Universidad Don Bosco

Vicerrectoría de Estudios de Postgrado

Maestría en Arquitectura de Software

Cohorte 2015

Tema:

"Propuesta de diseño de una API para un prototipo de hardware para realizar tracking de procesos de manufactura"

Presentado por:

Luis Josué Rodríguez Arguijo

Sandra Lisseth Carbajal Cruz

San Salvador, Septiembre de 2016

Contenido

Resumen

Este trabajo presenta un diseño de software Orientado a Objetos de una API para el control del avance del piso de producción y tiempos de mano de obra, que puede ser aplicado en empresas dedicadas a la manufactura. También presenta el diseño de un prototipo de hardware utilizando sistemas embebidos que se comunican con la API por medio de Servicios Web. El objetivo es brindar una propuesta de software y hardware que puedan ser aplicadas directamente en el piso de producción para controlar la efectividad de los procesos y servir como herramienta de mejora y toma de decisiones.

1 Introducción

La manufactura o producción, es una importante área en la economía de los países; y es necesario que las empresas de este rubro mejoren continuamente e innoven para ser competitivas.

El control es uno de los procesos que se ejecuta en la administración de las empresas, el cual sirve para medir la eficiencia de las operaciones, comparar el desempeño frente a lo planeado y para que la dirección tome medidas preventivas y correctivas.

En las empresas dedicadas a la manufactura es de vital importancia controlar los costos, los cuales implican principalmente la materia prima y la mano de obra. También es importante controlar que los pedidos para venta se satisfagan en tiempo, por lo cual es necesario monitorizar el trabajo en proceso y los avances de cada una de las órdenes de fabricación en el piso de producción.

Existen sistemas informáticos que ayudan a monitorizar el piso de producción, y las empresas pueden decidir si compran un sistema, como un ERP (Planeación de Recursos Empresariales); o desarrollan uno de acuerdo a sus necesidades.

Sin embargo, varios autores han planteado la diferencia o distanciamiento que hay entre los sistemas empresariales y el piso de producción. Muchos sistemas aunque permiten dar un seguimiento completo en planta, requieren que esta información sea digitada; otros necesitan que se utilice dispositivos de una marca determinada para poder integrarse con los sistemas existentes, y otros simplemente no permiten la integración con el piso de producción.

Con la idea de acercar los mecanismos que permitan medir los procesos de las plantas de producción algunas empresas optan por instalar computadoras, o comprar dispositivos de alto precio para que sus empleados capturen los datos requeridos para la gestión, tales como la estación de trabajo en que se encuentra, la entrada de las ordenes de fabricación, el estado de las órdenes, o las tareas ejecutadas por el operario; esto pude traer el inconveniente del uso

de espacio en la planta que puede afectar la movilidad o poner en riesgo algunos procesos, además de no usar de forma eficiente las capacidades de cálculo de los equipos instalados, usando sólo un porcentaje minúsculo de su capacidad de procesamiento.

Con la cada vez más alta densidad de transistores que se pueden integrar en una sola pastilla, se están desarrollando sistemas embebidos con capacidades de procesamiento superiores y a un costo relativamente bajo, con lo que es posible ejecutar tareas de poca demanda y en espacios cada vez menores facilitando su adaptación a las características particulares de las industrias de manufactura nacional.

En este artículo presentamos el diseño de una API con la cual se pretende dar una alternativa para el control de piso de producción, en lo referente al seguimiento del estado de las órdenes de fabricación en planta y la captura de información para el cálculo del costo de la mano de obra, además proponemos el diseño de un prototipo de hardware para la captura de estos datos desde el piso de producción usando tecnología de sistemas embebidos de hardware abierto.

El trabajo está organizado de la siguiente manera:

En la sección 2, se muestra el estado del arte y los trabajos relacionados con el control del piso de producción usando sistemas embebidos.

La sección 3, presenta los conceptos relacionados con la manufactura, que servirán para la especificación de requerimientos y el dominio de la aplicación; también se repasan los conceptos relacionados con microcontroladores y plataformas de prototipado.

La sección 4, muestra la metodología utilizada para la documentación de los requerimientos, el diseño de la solución de software (API), la selección de la placa de desarrollo usada y los proceso de construcción y pruebas del prototipo de hardware.

La sección 5, aborda el diseño de la API, que incluye los requerimientos planteados en formato de historias de usuario; se expone el diseño de la aplicación utilizando UML y la arquitectura del despliegue de la solución.

La sección 6, presenta el diseño del prototipo de hardware que se conectará a la API, la evaluación para seleccionar la placa de desarrollo a utilizar y cómo se realizará la comunicación desde el prototipo con la API.

Finalmente, la sección 7 presenta los resultados y ejecución de pruebas; y la sección 8 muestras las conclusiones del trabajo realizado.

2 Trabajos relacionados

El control de piso de taller y puntualmente el control de avances y prioridades en tiempo real, ha sido un tema en el que se han propuesto soluciones, tratando de superar los inconvenientes relacionados con el control manual de la producción y la integración con sistemas empresariales.

Kung-Yun propone un sistema en tiempo real basado en redes de sensores inalámbricos utilizando el protocolo Zigbee. Zigbee es una especificación para comunicación en red, de forma inalámbrica que es robusta, de bajo costo y que permite la creación de Redes Inalámbricas de Área Personal (WPAN). Se forma con dos tipos de dispositivos: los nodos, que tienen la capacidad de iniciar una red por sí solos; y los controladores que unen redes iniciadas por los nodos. Los dispositivos utilizados son de bajo consumo y tienen un rango de transmisión entre 10 y 100 metros (1).

Su solución implica la creación de una red en malla, con una computador central conectado por USB a un nodo Zigbee controlador, que permite la administración de la red de todos los dispositivos cliente, el almacenamiento y lectura en una base de datos MySQL o Access (2).

Cada dispositivo cliente en la red es capaz de leer etiquetas Zigbee que informan del avance de cada trabajo en proceso a medida se desplazan por las estaciones de trabajo. Para determinar el avance y la ubicación en planta, se localiza la etiqueta en base a algoritmos de proximidad en relación a los nodos instalados.

Esta solución sin embargo, necesita que el servidor sea conectado directamente al módulo controlador, lo cual impediría desplegar la solución en la nube si fuera necesario; así como la utilización del protocolo HTTP para la publicación de servicios.

También Moreira y CIA, han identificado la brecha existente entre los sistemas empresariales de back-end y los dispositivos que se utilizan en el piso de producción (3). En esta investigación propone la estandarización de dispositivos y sistemas empresariales a través de la utilización de Servicios Web[1] y la arquitectura SOA (Arquitectura Orientada a Servicios). El autor reconoce que aunque se implementen servicios tanto en el back-end como en los dispositivos, la implementación en estos últimos, es diferente en cuanto a la granularidad de las funciones que cumplen y su fiabilidad, sobre todo si la comunicación es inalámbrica. Aun así es posible integrar los dispositivos con soluciones empresariales, como ERP con módulos para el control de la producción.

[1] Servicios ofrecidos por un dispositivo electrónico a otro dispositivo electrónico a través de los protocolos de Internet.

Esta investigación aporta la propuesta de un framework para administrar dispositivos, permitir que se descubran de manera autónoma y puedan publicar y suscribirse a servicios web que proveen o consumen, ya sea con otros dispositivos o con aplicaciones ERP que cumplan con los requisitos, en este caso la solución utiliza SAP xMII, que es un portal de Inteligencia de Negocios orientado a la manufactura, comercializado por la empresa alemana SAP y que permite integrar la información de sistemas empresariales (4).

Sin embargo , esta solución requiere que los fabricantes de dispositivos utilizados en la manufactura reconozcan la factibilidad y los beneficios de incluir la capacidad de consumir y proveer Servicios Web embebidos en sus dispositivos. Caso contrario, no se podría alcanzar el nivel de estandarización requerido para su funcionamiento.

3 Marco conceptual

3.1 Manufactura

3.1.1 Definiciones de manufactura

La palabra *manufactura* se deriva de las palabras latinas *manus*(mano) y *factus*(hacer);la combinación de ambas significa *hecho a mano*. Esta frase hace referencia a los métodos manuales que se utilizaban en la antigüedad. La manufactura ha logrado una notable evolución, tanto que actualmente la mayoría de actividades se realizan con maquinaria automatizada controlada por computadora, requiriendo de supervisión manual.

La manufactura o fabricación es el proceso de conversión de materia prima en un producto específico, este proceso incluye el diseño del producto, la selección de la materia prima y la secuencia de procesos para crear el producto.

El diccionario de la Real Academia Española, define manufactura como una obra hecha a mano o con auxilio de máquina.

ElCAM-I (Consortium for Advanced Manufacturing), define la manufactura como una serie de actividades y operaciones interrelacionadas que involucran diseño del producto, maquinarias y herramientas.

Manufactura es el proceso de convertir materias primas en productos; y comprende las actividades en que el propio producto fabricado se utiliza para elaborar otros productos. Este autor sostiene que el nivel de manufactura de una nación se relaciona directamente con su salud económica, cuanto mayor es la actividad manufacturera de un país, mayor será el estándar de vida de sus habitantes (5).

En el contexto moderno se define la manufactura desde dos aspectos: tecnológico y económico (6).

Desde el aspecto tecnológico, manufactura es la aplicación de procesos físicos y químicos para alterar la geometría, propiedades o apariencia de un material de inicio dado para fabricar piezas o productos; la manufactura también incluye el ensamble de piezas múltiples para fabricar productos.

Desde el aspecto económico, manufactura es la transformación de los materiales en artículos de valor mayor por medio de una o más operaciones de procesamiento o ensamblado. El material se habrá hecho más valioso por medio de las operaciones de manufactura ejecutadas en él. Por ejemplo, cuando la arena se transforma en vidrio se le añade valor; cuando el petróleo se refina y se convierte en plástico su valor aumenta. La manufactura es el medio para crear bienestar material y es una actividad compleja que comprende una amplia variedad de recursos y actividades, entre ellas: Diseño del producto, Maquinaria y herramienta, Planeación del proceso, Materiales, Compra, Manufactura, Control de la producción, Servicios de soporte, Mercadeo, Ventas, Embarque y Servicios al cliente.

3.1.2 Historia de la manufactura

La palabra manufactura se deriva del latín manu factus, que significa "hecho a mano" y surgió por primera vez en 1567. Mientras que la palabra manufacturar apareció en 1683, posteriormente surge la palabra producto que significa "algo que se produce", junto a ella nació la palabra "producción", durante el siglo XV. Aunque “manufactura” y “producción” con frecuencia se utilizan de manera indistinta (5).

Sin embargo la manufactura se originó entre los años 5000 y 4000 a.C, es más antigua que la historia registrada. La manufactura de productos que tenían diversos usos específicos comenzó con la producción de artículos de madera, cerámica, piedra y metal.

En sus inicios la manufactura se desarrollaba completamente manual, con el paso del tiempo el hombre fue creando herramientas y utilizando nuevos materiales y operaciones más complejas que generaban una mayor capacidad de producción y un nivel mayor de calidad.

Los primeros materiales utilizados para fabricar utensilios domésticos y objetos ornamentales incluían metales como el oro, cobre e hierro, seguidos de la plata, el plomo, estaño, latón y bronce. La producción de acero (entre los años 600 y 800 d.C.) constituyó un hito importante; desde entonces se ha desarrollado una variedad muy amplia de metales ferrosos y no ferrosos. En la actualidad, los materiales que se emplean en productos avanzados, como computadoras y aeronaves supersónicas, incluyen materiales de ingeniería

(desarrollados para ese fin) con propiedades únicas, como cerámicos avanzados, plásticos reforzados, materiales compuestos y nano materiales.

Posteriormente se inventaron las maquinarias herramienta que en principio funcionaban con energía generada por vapor. Aproximadamente en1920, la electricidad había sustituido al vapor como la fuente principal de energía de las fábricas de Estados Unidos. Pero fue en el siglo XX donde sucedieron los mayores avances tecnológicos (6).

Durante la primera Revolución Industrial en 1750, los bienes se producían en lotes, lo cual requería de mucha confianza en la mano de obra en todas las fases de producción. Fue en el siglo XX, durante segunda Revolución Industrial, que inició el desarrollo de los dispositivos electrónicos de estado sólido y las computadoras, en esta época la mecanización comenzó en Inglaterra y se extendió en Europa y Estados Unidos, logrando un importante avance del diseño, la fabricación y el uso de partes intercambiables, creadas por Eli Whitney a principios de 1800. Con este avance, en la actualidad se puede reemplazar una pieza dañada por otra idéntica comprada en cualquier país (5).

Actualmente los métodos de producción han avanzado gracias al uso de los sistemas de manufactura integrados por computadora, utilizando micro y nano fabricación, por ejemplo, las latas de aluminio para bebidas se manufacturan a velocidades de 500 por minuto, los agujeros en las hojas metálicas se perforan a razón de 800 por minuto y las bombillas se elaboran en cantidades de más de 2000 por minuto.

3.1.3 Clases de industrias de manufactura

La manufactura se ejecuta como una actividad comercial de las compañías que venden productos a los clientes. El tipo de manufactura que realiza una industria dependerá del tipo de material a procesar, por ello hay industrias manufactureras: primarias, secundarias y terciarias. Las ***industrias primarias*** cultivan y explotan recursos naturales, tales como la agricultura y minería. Las ***industrias secundarias*** toman las salidas de las primarias y las convierten en bienes de consumo y capital. Las ***industrias terciarias*** constituyen el sector de servicios de la economía. Es así, que las salidas de una clase de industria, se convierten en entradas para la otra.

3.1.4 Tipos de productos manufacturados

Los productos manufacturados, se dividen en: bienes de consumo y bienes de capital (6). Los ***bienes de consumo*** son productos que los consumidores compran en forma directa, tales como autos, computadoras personales, televisores, etc. Los ***bienes de capital*** son aquellos

que adquieren otras compañías para producir bienes y prestar servicios, tales como aviones, equipo ferroviario, equipo de construcción, etc.

3.1.5 Capacidad de manufactura

Las industrias hacen uso de materiales, procesos, sistemas y personas para transformar dichos materiales en productos. Cada industria se especializa en la fabricación de productos específicos con determinados materiales, por ello nace la necesidad de medir la capacidad de manufactura.

La capacidad de producción, también conocida como capacidad de planta es la cantidad de producción que puede obtenerse en un periodo de tiempo dado, y se define como la tasa máxima de producción que una planta puede alcanzar en condiciones(número de turnos por semana, horas por turno) dadas de operación (6). Esta capacidad se mide en términos de unidades producidas, por ejemplo número de automóviles producidos por una planta de ensamblado final. (7), se refiere a este término como capacidad operativa, y es la cantidad de producción que un sistema es capaz de generar durante un período específico. Ambos autores coinciden en que para la planeación de la capacidad de producción, se deben considerar los factores tales como: capacidad tecnológica de proceso, limitaciones físicas del producto y capacidad de producción.

3.1.6 Operaciones de manufactura

Las operaciones dependerán del tipo de producto, pero las más comunes son operaciones de procesamiento y de ensamble.

Las operaciones de procesamiento se realizan con maquinaria y herramientas, también se requiere la intervención humana para controlar las maquinas, supervisar las operaciones tales como carga y descarga de las piezas antes y después de cada ciclo de operación. En algunos casos se requiere la ejecución de más de una operación de procesamiento para realizar la transformación del material y estas operaciones tienen una secuencia específica para lograr las especificaciones del diseño del producto.

El ensamble consiste en que dos o más piezas separadas se unen para formar una entidad nueva, la unión puede ser permanente o semipermanente. Este proceso se puede hacer de forma manual o con maquinaria especializada, dependiendo del tipo de ensamble.

A pesar del uso de máquinas y herramientas avanzadas, la mayoría de las operaciones de producción generan desperdicios y sobrantes, siendo el objetivo principal de toda manufactura recudir estos desperdicios. Para tal objetivo, algunas industrias utilizan sistemas informáticos, por ejemplo en una fábrica que procesa vidrio, un sistema de este tipo puede indicar cuáles son

los cortes más óptimos para generar menor desperdicio, esto a partir de una serie de entradas tales como el tipo de materia prima a utilizar, la cantidad y el tamaño de piezas a cortar.

3.1.7 Orden de fabricación

Una de las formas de controlar el proceso productivo es a través de órdenes de fabricación. La orden de fabricación detalla qué se debe fabricar, donde se debe fabricar, qué operaciones realizar (hoja de ruta o ruta de ensamble) y en qué fecha debe tener lugar la fabricación. También define cómo se deben liquidar los costes de la orden (8).

Una orden de fabricación puede generarse a partir de una demanda programada, es decir, cuando se fabrican productos de stock; o puede surgir a partir de una demanda actual, es decir, productos contra-pedido.

La orden de fabricación además debe definir la lista de materiales necesarios para llevar a cabo la producción, lo cual se denomina desglose de materiales o formulación. Dichos materiales pueden ser productos comprados a proveedores, lo cual puede generar una solicitud de compra; o, productos de fabricación propia, lo cual puede generar una o más órdenes de fabricación secundarias.

3.1.8 Líneas productivas

Es una serie de estaciones de trabajo ordenadas para que el producto pase de una estación a la siguiente y en cada ubicación se realice una parte del trabajo total. La velocidad de producción de la línea se determina por medio de su estación más lenta. Las estaciones de trabajo con ritmos más rápidos que el de la estación más lenta estarán limitadaspor este cuello de botella. Las líneas de producción se asocian con la producción masiva.

3.1.9 Estación de trabajo

Es el espacio físico donde se ejecuta uno o varios procesos para elaborar un producto, estos procesos pueden ser manuales o con maquinaria especializada.

3.1.10 Punto de control

En este trabajo, se le denominará punto de control al dispositivo que servirá para marcar el inicio y fin de un proceso de fabricación.

3.1.11 Sistemas de apoyo a la manufactura

La manufactura moderna desarrolla sus operaciones con maquinaria y herramientas avanzadas, incluyendo robots industriales. Además utiliza sistemas informáticos para apoyar las actividades realizadas en las siguientes áreas:

- Administración de la cadena de suministros: incluye todas las actividades relacionadas con la compra de materiales para la producción, gestión de proveedores, condiciones de compra de insumos, niveles de inventario, pronósticos de compra y colocación de pedidos en el punto de re orden adecuado.
- Administración de inventarios: controla la cantidad de materia actual, cantidad de productos en proceso y terminados, la recepción de pedidos, almacenamiento del material en las zonas correspondientes y el registro del inventario como parte de los bienes de la industria.
- Programación de la producción: se refiere al volumen de inventario, producción y subcontratación y a los tiempos de producción para satisfacer la demanda pronosticada establecida en el plan maestro.

3.1.12 Planeación y control de producción

La planeación de la producción, ayuda a determinar qué productos fabricar, qué cantidad, cuándo producirlas y con qué recursos. El control de producción, determina si ya se tienen los recursos para ejecutar el plan; sino, realiza las acciones para corregir esta deficiencia. El control de inventarios, es parte de la planeación y control de la producción, porque este se encarga de mantener los niveles de inventario adecuado de materia prima, trabajo en proceso y artículos terminados (6).

Los problemas en la planeación y control de la producción difieren en cada tipo de manufactura, para solucionar algunos problemas se requiere de una planeación detallada.

3.1.13 Sistema de planeación y control de producción

Toda compañía de manufactura debe tener un plan de negocios, en el que se debe incluir el tipo, la cantidad y el momento en que se fabricarán los productos. Este plan de manufactura debe incluir: pedidos actuales, pronósticos de venta, niveles de inventario y capacidad de la planta. Hay planes de manufactura a corto, mediano y largo plazo, y cada uno de estos tiene un responsable.

3.1.13.1 Plan agregado de producción

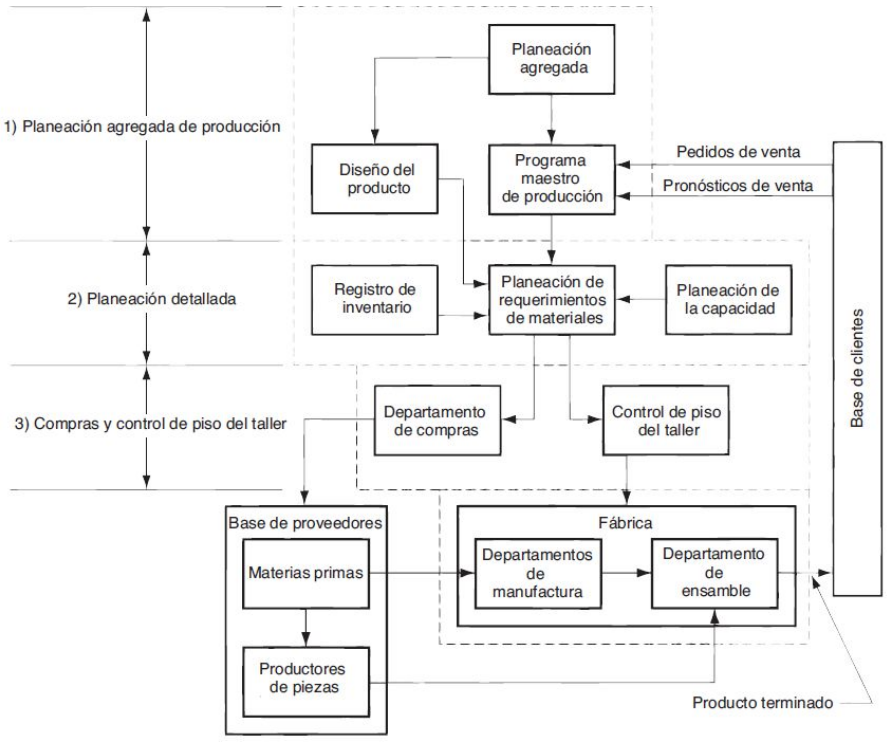
Es una planeación corporativa de alto nivel, en el que se integran los planes de ventas y mercadotecnia de la compañía, debe incluir los niveles actuales de inventario y los productos nuevos. En este plan se indican los niveles de resultados de producción para las principales líneas de productos y no para productos específicos.

3.1.13.2 Programa maestro de producción

Es un programa específico que enlista los productos que se van a fabricar, cuándo deben terminarse y en qué cantidades. Para ello, se apoya en:

- Pedidos de clientes: los pedidos obligan a la compañía a cumplir con una fecha de entrega.
- Demanda pronosticada: se refiere al nivel de resultados de la producción basados en datos históricos.
- Piezas de repuesto: se refiere a la solicitud de piezas, componentes individuales, partes para reparación que se almacenan en el departamento de servicio de la empresa.

Figura 1 Actividades en un sistema de planeación y control de la producción.



Nota. Actividades involucradas en la planeación y control de producción. Tomada de (6)

3.1.14 Sistema de Control de piso de taller

Esta fase se relaciona con la administración del trabajo en proceso, la autorización de órdenes de producción, control del avance de las órdenes y obtener información actualizada sobre el estado de los pedidos.

El control de piso de taller también conocido como Control de la actividad de producción (CAP), este se encarga de vigilar la actividad real de fabricación de un producto o la prestación de un servicio; priorizando el orden en que se ejecutan dichas actividades (7).

Los principales insumos de información que utilizan los sistemas CAP incluyen: pedidos recién liberados, estado de los pedidos actuales, información de ruteo, información del tiempo de espera y estado de los recursos usados en la producción. Otros insumos importantes son la cantidad, tipo y condición de los recursos para la producción tales como: personal, herramientas, capacidad de la maquinaria o equipo y los materiales.

3.1.14.1 Asignación de prioridades

Para la asignación de prioridades al trabajo que se ejecuta en un centro de trabajo, existen diversas reglas, entre ellas:

- Fecha de vencimiento: esta regla selecciona la tarea cuya fecha de vencimiento sea más próxima, a fin de ejecutarla primero.
- Tiempo de Procesamiento más Corto (TPC): este método permite priorizar de acuerdo con el tiempo de procesamiento para realizarlas, donde la tarea con el tiempo de procesamiento más corto se coloca en primer lugar de la lista.
- *Holgura total*: en esta regla se elige una tarea específica, luego se calcula el tiempo total necesario para realizar todas las operaciones restantes del trabajo en cuestión, y después el tiempo total que transcurrirá hasta que la tarea se venza. Al restar el tiempo de procesamiento total del tiempo total hasta el vencimiento se obtiene un valor denominado *holgura*.
- Holgura por operación: es una variante de la holgura total. De acuerdo con esta regla, la holgura total se divide entre el número de operaciones restantes. La tarea con menor holgura total por operación se programa primero. Esto ofrece más información que la regla de holgura total, por ejemplo, la holgura promedio de cada operación en lugar de la holgura total de la tarea completa.
- Primero en llegar, primero en ser atendido: es la regla que más utilizan las organizaciones de servicios, como bancos y tiendas minoristas, aunque muchas veces se debe a que no tienen alternativa. Esta regla suele ser percibida como justa, ya que la tarea que ingresa primero a la operación tendrá prioridad de ejecución.

3.1.14.2 Módulos de control de piso

El sistema de control de piso, se divide en tres módulos (6):

- Liberación de pedidos: este proceso genera los documentos necesarios para procesar una orden de producción en la fábrica. Estos documentos se denominan paquete del taller y se refieren a: la hoja de ruta, las requisiciones para comprar los materiales, las tarjetas de empleados para reportar el tiempo de mano de obra directa del pedido, las boletas de desplazamiento para autorizar el transporte de piezas a centros de trabajo en la ruta de producción, las listas de piezas requeridas para trabajos de ensamble. El uso de la tecnología de código de barras para vigilar el estado de una solicitud, hace innecesarios algunos de estos documentos en papel.
- Programación de pedidos, en esta etapa se asignan los pedidos de producción a las estaciones de trabajo en la fábrica. La programación de pedidos en el control de piso del taller enfrenta dos problemas en la planeación y control de la producción: carga de máquinas y secuenciación de actividades de trabajo.
- Progreso de los pedidos: en este módulo se recoge la información útil para administrar la producción por medio del monitoreo del estado de las órdenes, el trabajo en proceso y otros parámetros en la planta que indican el avance y rendimiento de la producción. Existen diversas técnicas para reunir estos datos, algunos requieren que los trabajadores registren los datos en formatos de papel, que posteriormente se integran con técnicas completamente automatizadas que no requieren participación humana. Entre los reportes están: reportes de estado de órdenes de trabajo, reportes de progreso y reportes de excepciones.

3.1.14.3 Uso de tecnología

El uso de la tecnología en los diferentes procesos de producción de bienes y en la ejecución de tareas administrativas, ha permitido una notable reducción en los tiempos de estas actividades y una reducción en desperdicios de material. Además posibilita la toma de decisiones con información en tiempo real. Por ejemplo se puede conocer cuántas órdenes están en proceso de producción, cual es el avance de las órdenes de producción en cada estación de trabajo, cual es el proceso o la estación de trabajo que consume más tiempo, etc. Por lo anterior, el objetivo de este artículo es diseñar una API para un prototipo de hardware que permita registrar los tiempos de una orden de producción en cada estación de trabajo, ya que esto facilitará el tracking o el rastreo de los procesos de manufactura. Además ayudará a determinar los costos de mano de obra de manera exacta y proveerá información real de los tiempos de producción para detectar mejoras y elaborar un plan maestro más acertado,

utilizando tiempos estimados de acuerdo a los tiempos reales registrados. Esta herramienta facilitará el control de la actividad de producción (CAP), es decir, permitirá vigilar la actividad real de fabricación de un producto (7).

3.2 Microcontroladores

3.2.1 Definiciones de microcontroladores

Un microcontrolador es un circuito integrado, en cuyo interior posee toda la arquitectura de un computador, esto es CPU, memorias RAM, EEPROM, y circuitos de entrada y salida (9).

"Los microcontroladores están conquistando el mundo. Están presentes en nuestro trabajo, en nuestra casa y en nuestra vida en general. Se pueden encontrar controlando el funcionamiento de los ratones y teclados de los computadores, en los teléfonos, en los hornos microondas y los televisores de nuestro hogar" (10).

Un microcontrolador, es un circuito integrado programable que contiene todos los componentes de un computador. Se emplea para controlar el funcionamiento de una tarea determinada y, debido a su reducido tamaño, suele ir incorporado en el propio dispositivo al que gobierna. Esta última característica es la que le confiere la denominación de controlador incrustado.

Un microcontrolador es un computador completo, aunque de limitadas prestaciones, que está contenido en el chip de un circuito integrado y se destina a gobernar una sola tarea. Un microprocesador por otra parte, es un circuito integrado abierto con el que se pueden interpretar instrucciones de propósito más general, es decir, no está enfocado en hacer una única tarea.

Los microcontroladores son usados ampliamente en la actualidad y, dada su versatilidad, pueden ser utilizados en una gran cantidad de proyectos, los más comunes son (11):

- Juegos y juguetes: las tiendas actuales están llenas de dispositivos que caminan, hablan, emiten luces e incluso reconocen la voz.
- Aplicaciones caseras: los electrodomésticos utilizan microcontroladores que realizan labores como establecer un temporizador en los hornos microondas, regular temperatura, velocidad de una licuadora, etc.
- Automatización de manufactura: para las empresas dedicadas a fabricar productos en grandes cantidades, es necesario auxiliarse con microcontroladores que asistan el proceso en tareas como dar seguimiento al inventario, testear para encontrar errores, entre otros.

- Detección del ambiente y respuesta a estímulos: los microcontroladores pueden ser ubicados en condiciones donde es impráctico o arriesgado utilizar humanos, como rastrear fugas de gas, medir temperatura, humedad del ambiente, entre otros.
- Automatización de edificios: los ingenieros diseñan edificios que ajustan calefacción y enfriamiento, abren y cierran puertas, hacen eficiente el uso de energía, etc.
- Control de procesos: en la industria los microcontroladores se han utilizado para controlar líneas de ensamblaje, por ejemplo, para verificar si todas las botellas en una fábrica de bebidas han sido llenadas con el nivel correcto y retirar de la líneas las que no cumplan esta característica.

Las partes principales de un microcontrolador son:

- El procesador.
- Memoria no volátil donde almacenar el programa embebido.
- Memoria de lectura y escritura para guardar datos.
- Controladores periféricos (comunicación en paralelo, serie y otras puertas de comunicación).
- Recursos auxiliares (reloj, temporizadores, conversores AC-DC, estado de reposo o bajo consumo).

Los microcontroladores están diseñados para que en su memoria de programa se almacene todas las instrucciones, por lo tanto, no es posible utilizar memorias externas de ampliación; y, como el programa a ejecutar es siempre el mismo debe quedar grabado de forma permanente.

Asimismo, procesan datos que varían conforme la ejecución del programa almacenado, para lo cual se necesita una memoria de lectura y escritura, en la mayoría de casos se utiliza memoria RAM estática (SRAM), la cual es volátil, aunque pueden tener también memoria de tipo EEPROM (no volátil) lo que permite no perder información con cortes de energía.

Para programar los microcontroladores se pueden utilizar lenguajes de bajo nivel (más cercanos al nivel de la máquina), los cuáles son más eficientes en espacio utilizado y ejecución. Sin embargo, pueden programarse con lenguajes de más alto nivel como C y BASIC, entre otros.

3.2.2 Definiciones de internet de las cosas

Gracias al progreso de la industria de semiconductores y la disminución de los precios, actualmente todas las partes de una computadora pueden ser puestas en un solo chip microcontrolador, el cual puede tener más capacidad que una computadora Apple II y correr 20

veces más rápido, y además puede comprarse por un precio aproximado a $10. Estos dispositivos pueden ser conectados a una red o a Internet (12).

El Internet de las cosas o Internet de los objetos consiste en que las cosas tengan conexión a Internet en cualquier momento y lugar. También se refiere a la integración de sensores y dispositivos en objetos cotidianos que quedan conectados a internet a través de redes fijas e inalámbricas (13).

Los microcontroladores y el Internet de las cosas permiten crear dispositivos especializados en tareas, y estos pueden transmitir dicha información por la red, sea local o internet. De esta manera, se pueden construir prototipos de hardware que utilicen sensores (dispositivos para capturar información del ambiente), o actuadores (dispositivos de salida, pantallas, motores, luces, etc.).

En el punto medio entre los microcontroladores y las computadoras personales se encuentran los Sistemas en Chips (SoC), que combinan procesador y los integran con distintos periféricos, teniendo capacidades de varios megabytes en RAM y utilizando memorias SD como medio de almacenamiento, un ejemplo de estos son la Raspberry Pi y el BeagleBone Black (14).

Al diseñar una solución basada en el Internet de las Cosas, se debe tomar una decisión sobre cuál es la plataforma de hardware más adecuada para dar solución a un problema concreto. Los siguientes son factores que se deben evaluar al momento de seleccionar una plataforma para un prototipo (14):

- Velocidad del procesador: indica que tan rápido puede procesar instrucciones individuales, puede evaluarse en relación a los millones de instrucciones por segundo (MIPS). También hay que tomar en cuenta si el procesador tiene soporte para cálculos de punto flotante[2]. Los microcontroladores suelen tener velocidades de decenas de MHz, mientras los Sistemas en Chip, cientos de MHz o algunos GHz.
- RAM: provee la memoria de trabajo para el sistema; en microcontroladores, debería contarse con al menos 4Mb de RAM; mientras en SoC, 256 o superior.
- Redes: cómo se conecta el dispositivo al resto del mundo es un factor clave; las opciones más comunes son Wifi, ZigBee, bluetooth, 3G u otras conexiones telefónicas.

[2] Forma de notación científica usada en los microcontroladores para representar números extremadamente grandes o pequeños de manera eficiente y compacta, y se pueden realizar operaciones aritméticas.

- USB: ¿Puede el dispositivo actuar como host y conectar otros periféricos como a una computadora? Con esto se puede ganar algunas características como almacenamiento adicional, adaptadores wifi, entre otros.
- Consumo de energía: los procesadores más rápidos consumen más energía que los más lentos; este es un punto a considerar si el prototipo dependerá de baterías o energía solar; y aunque utilice otro tipo de alimentación, siempre es deseable un bajo consumo. Algunos procesadores cuentan con un modo de reposo de bajo consumo.
- Interfaz con sensores y otros circuitos: la capacidad de manipular motores, LEDs, pantallas entre otros. Esto se logra a través de buses de comunicación como SPI o IC2.
- Tamaño y forma: El tamaño de la tarjeta influyen en el prototipo final; un procesador pequeño puede ser beneficioso, pero hace más difícil la tarea de soldar.

3.2.3 Definiciones de plataformas de hardware para prototipos

En esta sección se tratarán tres de las plataformas más destacadas. Se entiende por prototipo a un modelo utilizado para representar una idea de software y/o hardware con el propósito de poner a prueba nuevos conceptos de diseño para determinar su eficacia. Los prototipos pueden tener diferentes niveles de completitud (15).

Al diseñar un prototipo de hardware, se debe documentar las conexiones y componentes utilizados en este; para ello se utilizan los diagramas esquemáticos o diagramas de circuito, esta es la forma estándar de describir componentes y conexiones en un circuito eléctrico (16). Este tipo de diagrama utiliza símbolos para representar componentes y líneas para representar conexiones del circuito, aunque estas no representan la disposición o posición real de las conexiones físicas.

3.2.3.1 Arduino

Una plataforma de prototipado es Arduino, es un emblema en plataformas de Internet de las Cosas y computación física. Nació con el objetivo de cambiar el hecho que las tarjetas existentes eran costosas, difíciles de usar o ambas. Se decidió liberar el código y los esquemas y hacerlos de código abierto; esto permitió el nacimiento de un ecosistema de tarjetas y kits de trabajo. Se han creado varias versiones: Arduino NG, Diecimila, UNO, Due, entre otros.

El más popular es el Arduino Uno que posee un microcontrolador ATmega328, ranura USB para conectarse con la computadora, 32 Kb de almacenamiento, 2Kb de RAM y 14 puertos

GPIO[3]. Arduino fue diseñado pensando en la simplicidad, tanto en interconexión de componentes como en codificación y carga del programa. La codificación se realiza en un lenguaje derivado de C++ que permite manipular los puertos de entrada y salida, conocidos como pines, que pueden ser digitales; es decir, tomar únicamente valor de encendido o apagado; o, analógicos, que son valores identificados por el voltaje suministrado al pin.

3.2.3.2 Raspberry Pi

Otra plataforma de hardware es la Raspberry Pi, la cual es un SoC desarrollado con propósitos educativos y no para la computación física. La idea principal fue proveer una computadora que fuera pequeña y barata diseñada para ser programada y experimentada (17). Utiliza un chip Broadcom BCM2836, tiene capacidad avanzada de gráficos de alta definición y un CPU de 700 MHz o superior en nuevas versiones.

Utiliza una distribución de Linux llamada Raspbian, que es una variante de Debian, el cual es instalado en una memoria micro SD. Es necesario aclarar que aunque el Raspberry Pi de menos capacidad es muy superior al mejor de los Arduinos, en cuanto a sus características, la capacidad para controlar el mundo físico a través de sensores y actuadores es muy diferente. A pesar de ello, también cuenta con pines GPIO programables desde Python, por ejemplo, que es el lenguaje recomendado (14).

3.2.3.3 Beaglebone Black

Similar a la Raspberry Pi es la Beaglebone Black, una tarjeta fabricada por Beaglebone Group, una empresa conformada en gran parte por empleados de Texas Instruments, creada con el objetivo de facilitar la educación en electrónica con dispositivos incrustados poderosos y abiertos. La tarjeta original no tiene video ni audio, pero trae un gran número de pines GPIO. La Beaglebone fue lanzada antes que la Raspberry Pi, y el lanzamiento de esta última hizo que se revisara la plataforma y se bajaran los costos a la mitad al crear la Beaglebone Black, que aunque sigue sin tener audio y video analógico, posee un conector micro-HDMI.

Esta tarjeta también corre varias distribuciones de sistemas operativos Linux, pero no fue diseñada para comunicarse con el usuario final vía pantalla y teclado.

---

[3]GPIO, Puertos de Propósito General de Entrada y Salida.

3.3 API

3.3.1 Definiciones de API

API es la forma corta de referirse a una Interfaz de Programación de Aplicaciones (Application Programming Inteface, en inglés). El término API se refiere a clases, interfaces, constructores o miembros por los cuales un programador accede a una clase, interfaz o paquete. Este término es utilizado en lugar de simplemente interfaz para evitar confusión con las interfaces utilizadas en programación orientada a objetos para implementar el polimorfismo (18).Se conoce como usuario de una API al programador que escribe código que utiliza una API; mientras que la clase que la implementa es conocida como cliente de la API.

Las clases, interfaces, constructores y miembros que la componen son conocidos como elementos de la API. La API consiste de elementos que son accesibles desde afuera del paquete que la define; es decir, la API de un paquete consiste en los miembros y constructores públicos y protegidos de cada clase pública o interfaz del paquete.

Existen APIs que pueden ser accedidos desde clientes a través de servicios web, estas se denominan APIs web. Un servicio web es una entidad programable que provee elementos de funcionalidad, como lógica de aplicaciones, y que es accesible a cualquier número de sistemas potencialmente dispares utilizando estándares de Internet, tales como XML y HTTP (19).

Los servicios web son servidores que apoyan las necesidades de un sitio u otra aplicación (20). Una API expone un conjunto de datos y funciones para facilitar la interacción entre programas de computadora y permitirles intercambiar información: una API web es el rostro de un servicio web, directamente escuchando y respondiendo solicitudes de clientes. Una API web que cumpla el estilo arquitectónico de REST es una API REST.

En 1993, Roy Fielding, co-fundador del proyecto Apache estudió el problema de la escalabilidad de la web, y determinó que para que la web se expandiera con éxito, era necesario que lograra satisfacer un conjunto de restricciones clave, como la separación de funciones entre cliente y servidor, la uniformidad y estandarización de interfaces, manejo del caché, código a demanda, entre otros. Su trabajo lo llevó a escribir una nueva versión del protocolo HTTP, su especificación y la formalización de los Identificadores Uniformes de Recursos (URI). A este estilo de arquitectura le llamó Transferencia de Estado Representacional (REST), que está compuesta por las restricciones anteriores.

4 Metodología

La metodología es el conjunto de procedimientos ordenados que nos permitirá alcanzar los objetivos planteados. En la tabla 1, se presentan los objetivos del proyecto y la metodología para lograr dichos objetivos.

Tabla 1 Matriz de congruencia para el desarrollo del proyecto.

Problema General	**Objetivo General**	**Metodología**
¿Cómo elaborar una API para un prototipo de hardware para realizar tracking de procesos de manufactura?	Diseñar una API para un prototipo de hardware para realizar tracking de procesos de manufactura.	**Selección del hardware y sus componentes:** • Criterios ponderados. **Diseño y construcción del prototipo de hardware:** • Diseño esquemático del hardware. • Especificaciones técnicas de los componentes. • Criterios de evaluación de componentes de hardware. • Historial de pruebas realizadas. **Diseño y desarrollo del software:** Se utilizará el lenguaje de modelado UML, la cual tendrá como resultado la siguiente documentación: • Especificación de requerimientos con historias de usuario. • Modelo de dominio • Diagrama de clases • Diagrama de secuencia • Diagrama de componentes y despliegue. **Planificación y construcción del software:** • SCRUM
Problemas específicos	**Objetivos Específicos**	
¿Cómo elaborar un prototipo de hardware para realizar tracking de procesos de manufactura?	Diseñar un prototipo de hardware con sus componentes y periféricos necesarios para realizar el tracking de las órdenes de fabricación.	
¿Cómo debe ser el software embebido que utilizará el prototipo de hardware?	Diseñar el software embebido en el prototipo de hardware.	
¿Cómo debe ser el diseño de una API que permita la comunicación entre el prototipo de hardware y otros sistemas de tracking de procesos de manufactura?	Diseñar una API que permita la comunicación entre el prototipo de hardware y otros sistemas en los que se registren los datos del tracking de procesos de manufactura.	

Nota. Matriz de congruencia para el desarrollo del proyecto. Elaboración propia.

4.1 Metodología para la selección del hardware y sus componentes

4.1.1 Criterios ponderados

Se utilizará un modelo para la selección del hardware y componentes del prototipo, construido a partir de los factores planteados (14), estos factores están descritos en la sección *Definiciones de Internet de las cosas* dentro marco conceptual.

4.1.1.1 Criterios de evaluación para la tarjeta microcontroladora

Los criterios de evaluación para la tarjeta microcontroladora, se muestran en la tabla 3, esta contiene las siguientes columnas: los factores a evaluar, el peso asignado a cada uno, el puntaje alcanzado por cada opción evaluada y el valor porcentual de dicha evaluación.

El peso de los criterios se definirá de acuerdo a fuentes bibliográficas, especificaciones de fabricantes y el criterio de los investigadores, este tipo de fuentes también se utilizará para asignar el puntaje a cada opción. Sin embargo, el resultado de la evaluación se validará con la opinión de expertos, con el objetivo de identificar las principales desventajas que el modelo seleccionado podría presentar, de acuerdo a proyectos desarrollados.

La escala para asignar el puntaje constará de 5 valores, mostrados en la tabla 2..

Tabla 2 Criterios de evaluación de tarjeta microcontroladora

Puntaje	Significado
1	No cumple con el criterio.
2	Funcionalidad incompleta.
3	Cumple el requisito mínimo.
4	Cumple el requisito completo.
5	Valor añadido.

Tabla 3.a Criterios de evaluación de tarjeta microcontroladora

Criterio	Descripción	Peso	Puntaje Opc1	Puntaje Opc 2	Opc 1 %	Opc 2 %
Procesador	Velocidad del microprocesador, MIPS (millones de instrucciones por segundo).					
RAM	Cantidad de memoria RAM disponible para procesar los datos.					
Facilidad de uso	Facilidad de aprendizaje del lenguaje de programación y ensamblado de los componentes.					
Consumo de energía	Permite en conjunto con los demás componentes no exceder los 20 Voltios y 3.5 Amperios.					
USB	Capacidad para conectar un dispositivo USB al microcontrolador.					
Facilidad de conexión y ensamble	Es fácil de ensamblar y conectar con las demás partes del prototipo.					

Nota. Detalle de criterios de evaluación para seleccionar la tarjeta microcontroladora para el prototipo. Elaboración propia a partir de (14).

Tabla 3.b Criterios de evaluación de tarjeta microcontroladora

Criterio	Descripción	Peso	Puntaje Opc1	Puntaje Opc 2	Opc 1 %	Opc 2 %
Costo	Mantiene el valor del prototipo relativamente bajo (por definir presupuesto)					
Tamaño	Ayuda a mantener compacto el tamaño del prototipo.					
Presencia en el mercado nacional	Existen distribuidores locales de la tarjeta microcontroladora.					
Conectividad a red	Permite la conexión a red local o Internet de forma cableada e inalámbrica.					
Madurez en el mercado	Cantidad de años en el mercado					
Comunidad y soporte activo	Se encuentra soporte y foros activos en las páginas oficiales					

Nota. Detalle de criterios de evaluación para seleccionar la tarjeta microcontroladora para el prototipo. Elaboración propia a partir de (14).

4.1.1.2 Criterios de evaluación de componentes

Tabla 4 Criterios de evaluación de componentes

Criterio	Descripción	Peso	Puntaje Opc. 1	Puntaje Opc. 2	Opc. 1 %	Opc 2 %
Estabilidad	Funciona de manera estable sin sufrir desconexiones, pérdida de información, fluctuaciones, reinicios o congelamientos.					
Facilidad de conexión y ensamble	Es fácil de ensamblar y conectar con las demás partes del prototipo.					
Facilidad de uso	El componente no representa una dificultad extra para su uso. Buena visualización y captura de los datos requeridos.					
Consumo de energía	Permite en conjunto con los demás componentes no exceder los 20 Voltios y 3.5 Amperios					
Costo	Mantiene el valor del prototipo relativamente bajo (por definir presupuesto)					
Tamaño	Ayuda a mantener compacto el tamaño del prototipo.					
Estética	Es presentable y elegante a la vista.					
Resistencia	No necesita ser tratado con cuidado o delicadeza excesiva.					
	TOTAL	**100**				

Nota. Detalle de criterios de evaluación para seleccionar los componentes para el prototipo. Elaboración propia a partir de (14).

La tabla 4, contiene los factores a evaluar para la selección de componentes. Algunos de estos factores son similares a los de la tarjeta microcontroladora. Para asignar el peso de los indicadores y el puntaje de cada opción evaluada, se utilizarán fuentes bibliográficas, especificaciones de fabricantes y el criterio de los investigadores. La escala de valores para definir el puntaje es la que se detalló en la tabla 2.

4.1.2 Perfil de experto para validación de la selección de la tarjeta microcontroladora

Luego de realizar la evaluación de la tarjeta microcontroladora y elegir la opción que resulte mejor evaluada, se consultará con expertos la valoración que tienen sobre la tarjeta, experiencias previas o actuales, sugerencias de diseño y posibles problemas a enfrentar en la elaboración del prototipo y su interacción con la API de comunicación. El perfil de experto para dicha validación se muestra en la tabla 5.

Tabla 5 Perfil de experto para selección de plataforma de hardware

Característica	Descripción
Formación académica	Graduado de ingeniería eléctrica, electrónica o carreras afines, con conocimiento y experiencia en implementación de proyectos con microcontroladores.
Experiencia con la tarjeta microcontroladora seleccionada	Requerida.
Tiempo de experiencia	Un año.

Se aplicará una entrevista semiestructurada con los expertos seleccionados. El instrumento utilizado será un cuestionario con las siguientes preguntas como base para su aporte:

- ¿Cuáles son ventajas que se pueden explotar con la tarjeta, que a su criterio, se puedan aplicar al prototipo a desarrollar?
- ¿Qué problemas y limitaciones identifica en la placa seleccionada que pueden perjudicar o complicar la elaboración del prototipo?
- ¿Qué forma de conexión a red es la más adecuada al utilizar la placa seleccionada?
- ¿Qué consideraciones en cuanto a la alimentación eléctrica se deben tener para la realización del prototipo?
- ¿Qué componentes de hardware ha utilizado o recomienda para la elaboración del prototipo (Display, teclado, lector de barras, etc.)?.

4.2 Metodología para la construcción del prototipo

Tabla 6 Formato para historial de pruebas

ID	Fecha	Descripción	Categoría	Éxitos	Fallos	Siguiente Prueba
1	01/01/2016	Conexión del dispositivo a red cableada utilizando DHCP. Componentes: Microcontrolador. Router marca X. Cable de alimentación de 5v AC/DC	RED	El dispositivo obtuvo una dirección IP dinámica y logró hacer PING desde la máquina de prueba con dirección 192.x.x.x	La conexión a red se interrumpió después de N minutos.	Verificar la configuración del componente de red del dispositivo.

Tabla 7 Descripción del formato de historial de pruebas

Columna	Descripción del contenido
ID	Correlativo de la prueba
Fecha	Fechas en las que se realizó la prueba. Puede ser un rango si la prueba abarca varios días.
Realizado por	Nombre de la persona que realizó la prueba.
Descripción	Descripción de la prueba, narración del procedimiento ejecutado, componentes involucrados y resultados esperados.
Categoría	Red=Pruebas de conectividad a la red. Display=Prueba de display de datos. Input=Prueba de digitación y captura de datos. Software=Pruebas del programa que controla el dispositivo. Eléctrico=Pruebas de estabilidad y consumo eléctrico, desempeño según alimentación eléctrica usada.
Éxitos	Descripción de los logros realizados y los objetivos alcanzados y su grado de alcance (parcial y completo).
Fallos	Descripción de los objetivos no logrados y resultados no favorables derivados de la prueba.
Siguiente Prueba	Descripción de la siguiente prueba a realizar o las variantes que se deben considerar. Agregar el ID de referencia donde se ejecutó la siguiente prueba.

La elaboración del prototipo requiere la realización de pruebas de componentes, estabilidad, conectividad, entre otros. Por ello, es necesario llevar un control detallado de todas las pruebas realizadas, definiendo si fue exitosa o no y documentar las lecciones aprendidas para futuros diseños. También es importante definir si la prueba concluyó exitosamente y

formará parte del prototipo final. La tabla 6 muestra el formato que se utilizará para registrar el historial de pruebas; mientras que la tabla 7 contiene los detalles de lo que se debe complementar en cada columna.

El prototipo será diseñado por medio de un diagrama esquemático o diagrama de circuito, que especificará las conexiones entre componentes y la tarjeta microcontroladora. Se utilizará un software gratuito para el diseño de circuitos.

4.3 Metodología para el diseño y desarrollo del software

Existen diversas metodologías para planificar y controlar el proceso de desarrollo de software. Sin embargo, para el diseño propuesto se utilizará la siguiente metodología:

4.3.1 UML

La documentación para el diseño de la API y el software de prueba incluye lo siguiente:

- Especificación de requerimientos con historias de usuario.
- Modelo de dominio.
- Diagrama de clases.
- Diagrama de secuencia.
- Diagrama de componentes y despliegue.

4.3.1.1 Historias de usuario

Tabla 8 Formato para documentar una historia de usuario

Nombre			
Id		**Importancia**	
Rol		**Estimación**	
Enunciado			
Criterios de aceptación			

Nota. Elementos que debe contener una historia de usuario. Elaboración propia a partir de (21)

En las metodologías ágiles la descripción de las necesidades se realiza a partir de las historias de usuario, ya que estas se enfocan en definir lo que el usuario necesita hacer, sin describir el cómo. Estas contienen dos elementos importantes: enunciado y criterio de aceptación. El concepto de historia de usuario tiene sus raíces en la metodología

“eXtremeProgramming” o programación extrema. Esta metodología fue creada por Kent Beck y descrita por primera vez en 1999 en su libro “eXtreme Programming Explained”. Sin embargo, las historias de usuario se adaptan de manera apropiada a la mayoría de las metodologías ágiles, una de ellas es Scrum.

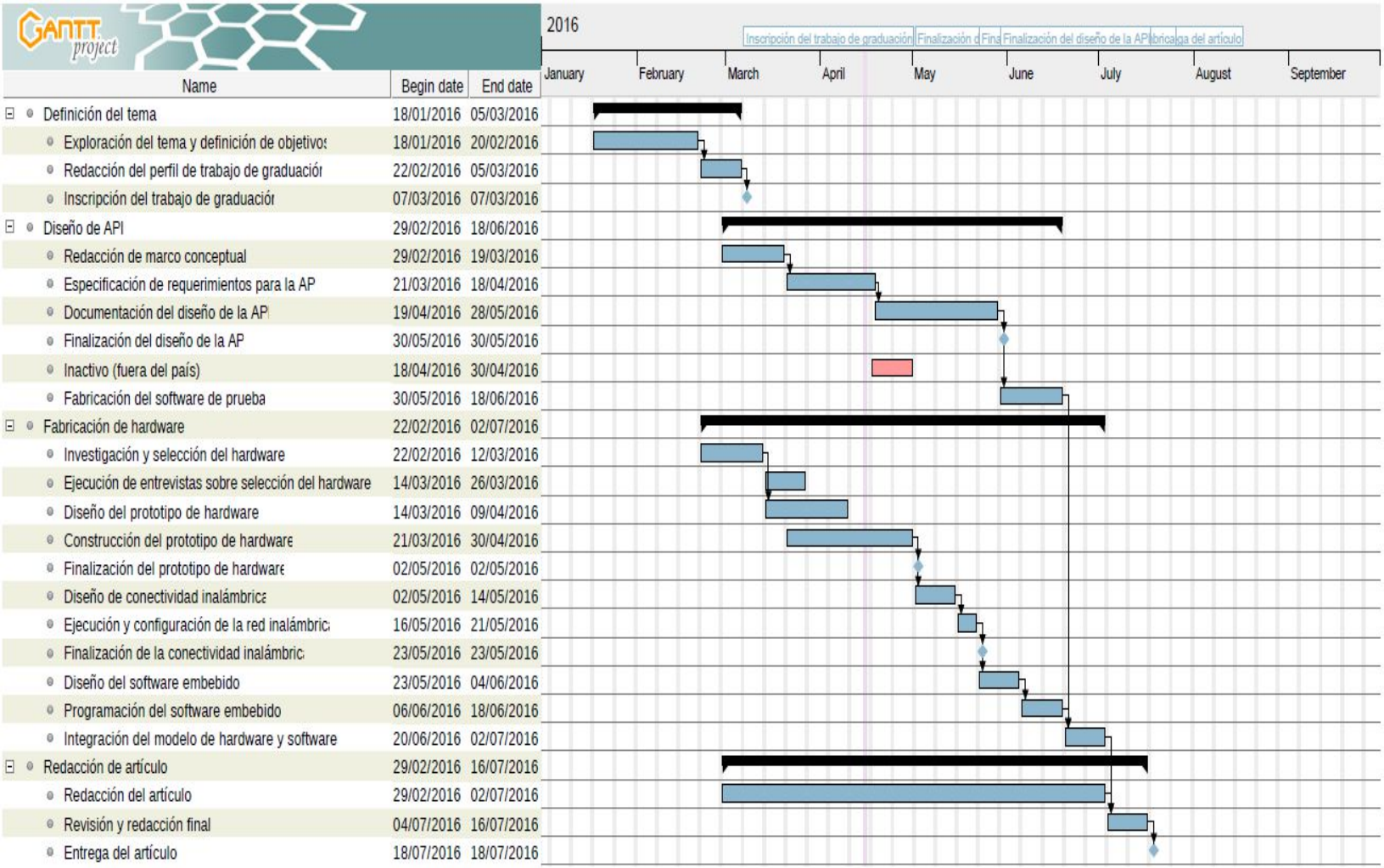
En la tabla 8 se presenta el formato que se utilizará para registrar las historias de usuario para el proyecto a desarrollar.

4.3.2 SCRUM

Este modelo fue diseñado por Ikujiro Nonaka e Hirotaka Takeuchi a principios de los 80, al analizar cómo desarrollaban los nuevos productos las principales empresas de manufactura tecnológica. SCRUM es un marco de desarrollo ágil caracterizado por adoptar una estrategia de desarrollo incremental, en lugar de planificar y desarrollar un producto completo. Este tipo de estrategia permite cumplir las expectativas del cliente, lograr una mayor flexibilidad ante los cambios del producto, realizar predicción de tiempos, conseguir mayor productividad en el equipo, entre otras.

Para la gestión del desarrollo de la API, software embebido y software de pruebas, se utilizará SCRUM, ya que este permite realizar entregas parciales y regulares del producto final facilitando el trabajo en equipo y la revisión del producto contra los requerimientos.

4.4 Cronograma



Name	Begin date	End date
Definición del tema	18/01/2016	05/03/2016
Exploración del tema y definición de objetivos	18/01/2016	20/02/2016
Redacción del perfil de trabajo de graduación	22/02/2016	05/03/2016
Inscripción del trabajo de graduación	07/03/2016	07/03/2016
Diseño de API	29/02/2016	18/06/2016
Redacción de marco conceptual	29/02/2016	19/03/2016
Especificación de requerimientos para la API	21/03/2016	18/04/2016
Documentación del diseño de la API	19/04/2016	28/05/2016
Finalización del diseño de la API	30/05/2016	30/05/2016
Inactivo (fuera del país)	18/04/2016	30/04/2016
Fabricación del software de prueba	30/05/2016	18/06/2016
Fabricación de hardware	22/02/2016	02/07/2016
Investigación y selección del hardware	22/02/2016	12/03/2016
Ejecución de entrevistas sobre selección del hardware	14/03/2016	26/03/2016
Diseño del prototipo de hardware	14/03/2016	09/04/2016
Construcción del prototipo de hardware	21/03/2016	30/04/2016
Finalización del prototipo de hardware	02/05/2016	02/05/2016
Diseño de conectividad inalámbrica	02/05/2016	14/05/2016
Ejecución y configuración de la red inalámbrica	16/05/2016	21/05/2016
Finalización de la conectividad inalámbrica	23/05/2016	23/05/2016
Diseño del software embebido	23/05/2016	04/06/2016
Programación del software embebido	06/06/2016	18/06/2016
Integración del modelo de hardware y software	20/06/2016	02/07/2016
Redacción de artículo	29/02/2016	16/07/2016
Redacción del artículo	29/02/2016	02/07/2016
Revisión y redacción final	04/07/2016	16/07/2016
Entrega del artículo	18/07/2016	18/07/2016

5 Diseño de la API

5.1 Requerimientos

En esta sección, se listan las necesidades clave que debe cumplir el diseño de la API para realizar el tracking de procesos de manufactura, específicamente para órdenes de fabricación (OF). Dichos requerimientos surgen de la literatura de expertos y del conocimiento que poseen los investigadores en el área de manufactura.

5.1.1 Requerimientos de software

El software de tracking de órdenes de fabricación deberá:

- Dar seguimiento a las órdenes de fabricación en cada uno de los procesos definidos en la ruta de ensamble de dicha OF.
- Marcar el inicio del trabajo de la orden de fabricación en cada proceso.
- Definir pausas en el trabajo de las órdenes de fabricación.
- Marcar como finalizada una orden de fabricación para permitir su procesamiento en el siguiente punto de la ruta de ensamble.
- Validar que la orden de fabricación ingresada este registrada y en proceso (no finalizada).
- Validar que el proceso que registre tiempo a la orden de fabricación sea parte de la ruta de ensamble de dicha OF.
- Permitir pausa de una OF si ya está iniciada, registrando el tiempo en que se realizó la pausa.
- Permitir el ingreso de una OF a bodega cuando ya se han registrado los tiempos de todos los procesos según la ruta de ensamble.
- Permitir el ingreso de productos a bodega validando contra el detalle de artículos de la OF.
- Validar que la cantidad de artículos ingresados no sobrepase la cantidad que solicite la OF, incluyendo los ingresos parciales realizados anteriormente.
- Permitir la captura de datos por medio del hardware y tener disponibles estos datos para que sean utilizados en otros sistemas por medio de una API.
- Brindar información sobre cuáles son las órdenes de fabricación que se encuentran en la cola de trabajo de cada proceso.
- Permitir la medición de la cantidad de tiempo asignado en cada proceso productivo a cada orden de fabricación.

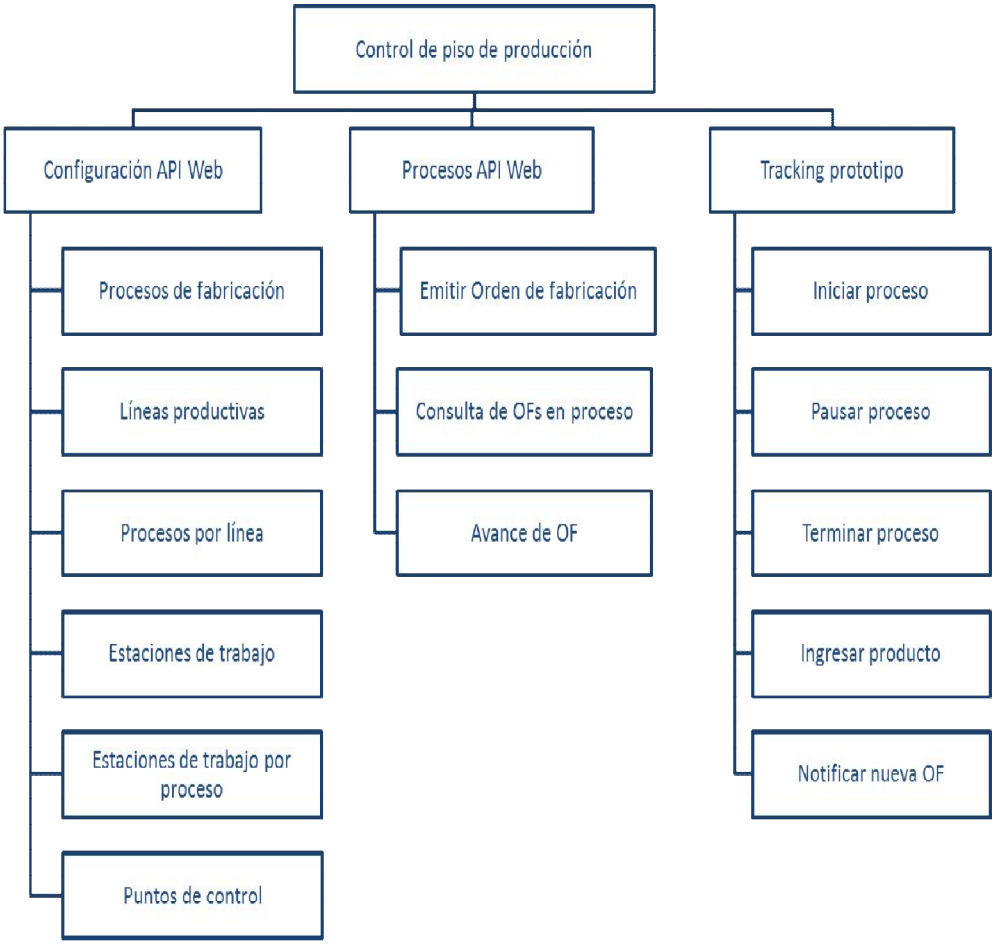
- Asignar a cada dispositivo el proceso productivo al que pertenece.

5.1.2 Historias de usuario

En este apartado, se detallan los requerimientos de software usando historias de usuario. Recordando que una historia de usuario es el "qué" quiere el usuario y el caso de uso es un "cómo" lo quiere. Posteriormente, estas historias de usuario, se complementarán con casos de uso donde una historia de usuario puede dar lugar a la especificación de varios casos de uso.

En la Figura 2, se muestran las historias de usuario de acuerdo a los requerimientos de software, teniendo en cuenta que estos incluyen únicamente requerimientos funcionales que pueden ser realizados desde la API o desde el prototipo de hardware propuesto.

Figura 2 Historias de usuario



A continuación, se detalla cada historia de usuario identificada en la sección anterior:

5.1.2.1 Registrar los procesos de fabricación.

1	**Registrar los procesos de fabricación.**	
Estimación:	**Valor:**	**Dependencias:**
Descripción		
Como [planificador de producción], quiero [crear un catálogo de los procesos de fabricación], para [la creación de líneas productivas].		
Criterios de aceptación		
• Ingresar los datos del proceso de fabricación, si este ya está registrado, mostrar mensaje de error. • Introducir los datos del proceso de fabricación y almacenarlo en caso que no esté registrado. • Consultar los procesos registrados en el sistema y permitir la modificación de estos.		

5.1.2.2 Registrar las líneas productivas.

2	**Registrar las líneas productivas.**	
Estimación:	**Valor:**	**Dependencias:**
Descripción		
Como [planificador de producción], quiero [crear un catálogo de líneas de producción para [registrar las líneas productivas].		
Criterios de aceptación		
• Ingresar los datos de la línea productiva, si esta ya está registrada, mostrar mensaje de error. • Introducir los datos de la línea productiva y almacenarla en caso que no esté registrada. • Consultar las líneas registradas en el sistema y permitir la modificación de estas.		

5.1.2.3 Configurar los procesos de fabricación en cada línea productiva.

3	**Configurar los procesos de fabricación en cada línea productiva.**	
Estimación:	**Valor:**	**Dependencias:**
Descripción		
Como [planificador de producción], quiero [configurar los procesos de fabricación a realizar en cada línea productiva], para establecer [un orden en los procesos].		
Criterios de aceptación		
• Consultar las líneas productivas • Seleccionar los procesos de fabricación y asociarlos a cada línea productiva. • Validar que un proceso productivo no se repita en la misma línea.		

5.1.2.4 Registrar las estaciones de trabajo.

4	**Registrar las estaciones de trabajo.**	
Estimación:	**Valor:**	**Dependencias:**
Descripción		
Como [planificador de producción], quiero [crear las estaciones de trabajo], para registrar [los procesos de fabricación que se realizan en cada una de estas].		
Criterios de aceptación		
• Ingresar los datos de la estación de trabajo, si esta ya está registrada, mostrar mensaje de error. • Introducir los datos de la estación de trabajo y almacenarla en caso que no esté registrada. • Consultar las estaciones de trabajo en el sistema y permitir la modificación de estas.		

5.1.2.5 Configurar las estaciones de trabajo por proceso de fabricación.

5	**Configurar las estaciones de trabajo por proceso de fabricación.**	
Estimación:	**Valor:**	**Dependencias:**
Descripción		
Como [planificador de producción], quiero [asociar a cada estación de trabajo los procesos de fabricación que se realizan], para facilitar [la administración del proceso de fabricación].		
Criterios de aceptación		
• Consultar las estaciones de trabajo. • Seleccionar los procesos de fabricación y asociarlos a la estación de trabajo. • Validar que un proceso productivo no se repita en la estación de trabajo.		

5.1.2.6 Registrar los puntos de control.

6	**Registrar los puntos de control.**	
Estimación:	**Valor:**	**Dependencias:**
Descripción		
Como [planificador de producción], quiero [registrar los puntos de control], para disponer [de la información de estos].		
Criterios de aceptación		
• Ingresar los datos del punto de control, si este ya está registrado, mostrar mensaje de error. • Introducir los datos del punto de control y almacenarlo en caso que no esté registrado. • Consultar los puntos de control registrados en el sistema y permitir la modificación de estos.		

5.1.2.7 Configurar los puntos de control por estación de trabajo.

7	**Configurar los puntos de control por estación de trabajo.**	
Estimación:	**Valor:**	**Dependencias:**
Descripción		
Como [planificador de producción], quiero [configurar los puntos de control que se utilizarán en cada estación de trabajo], para poder [registrar los tiempos reales empleados en cada proceso al elaborar un determinado producto].		
Criterios de aceptación		
• Consultar los puntos de control. • Seleccionar un punto de control y asignarlo a la estación de trabajo donde se utilizará, validando que no esté asignado a otra estación.		

5.1.2.8 Emitir la orden de fabricación.

8	**Emitir la orden de fabricación.**	
Estimación:	**Valor:**	**Dependencias:**
Descripción		
Como [planificador de producción], quiero [generar la orden de fabricación], para poder [iniciar el proceso de producción de esta].		
Criterios de aceptación		
• Ingresar la fecha estimada para realizar la orden de fabricación. • Ingresar el detalle de artículos a producir y las especificaciones técnicas para fabricarlos. • Ingresar detalle de materias primas a utilizar. • Generar hoja de ruta.		

5.1.2.9 Iniciar proceso de fabricación.

9	**Iniciar proceso de fabricación.**	
Estimación:	**Valor:**	**Dependencias:**
Descripción		
Como [operario], quiero [marcar el inicio de cada proceso de una OF], para poder [registrar la hora real en que se inicia la producción].		
Criterios de aceptación		
• Ingresar el identificador del empleado que trabajará la OF, y validar que el empleado sea válido. • Ingresar el número de la OF por medio del teclado o del escáner. • Validar que el punto de control pertenezca a ruta de la OF. • Si la OF no está registrada en el sistema, mostrar mensaje de error. • Si el empleado ya tiene registrado un trabajo en el mismo (u otro) punto de control, debe pausar dicho trabajo e iniciar el que se está ingresando. • Si la OF está emitida o en proceso, permitir el inicio del proceso y almacenar la hora de inicio.		

5.1.2.10 Pausar proceso de fabricación.

10	**Pausar proceso de fabricación.**	
Estimación:	**Valor:**	**Dependencias:**
Descripción		
Como [operario], quiero [pausar un proceso fabricación], para poder [iniciar otro o indicar una pausa temporal en las labores].		
Criterios de aceptación		
• Ingresar el identificador del empleado que trabajará la OF, y validar que el empleado sea válido. • Si hay un proceso iniciado permitir la pausa, el sistema debe registrar la hora de dicha pausa.		

5.1.2.11 Terminar proceso de fabricación.

11	**Terminar proceso de fabricación.**	
Estimación:	**Valor:**	**Dependencias:**
Descripción		
Como [operario], quiero [terminar un proceso], para marcar [el tiempo real de ejecución de cada proceso productivo].		
Criterios de aceptación		
• Ingresar el número de la OF por medio del teclado o del escáner, y validar que el punto de control pertenezca a la ruta de la OF. • Si la OF no está iniciada, mostrar un mensaje de error. • Si la OF ya está iniciada y el proceso a terminar no es el último, se debe finalizar únicamente el proceso actual. • Si la OF ya está iniciada, y todos sus procesos ya están finalizados incluyendo el ingreso completo a bodega, permitir la finalización de la OF.		

5.1.2.12 Ingresar producto a bodega.

12	**Ingresar producto a bodega**	
Estimación:	**Valor:**	**Dependencias:**
Descripción		
Como [operario], quiero [ingresar a bodega los productos terminados en una OF], para disponer [de producto para la venta o producción de otra OF].		
Criterios de aceptación		
• Ingresar el código de empleado. • Ingresar el número de la OF por medio del teclado o del escáner, y validar que el punto de control pertenezca a ruta de la OF. • Permitir ingreso total o parcial, para ello ingresar los parámetros: Empleado, OF, articulo y cantidad. • Se debe validar los productos y cantidad a ingresar contra el detalle de la orden de fabricación. • Si ya se realizó el ingreso completo, el sistema debe terminar la OF.		

5.1.2.13 Notificar nueva OF.

13	**Notificar nueva OF.**	
Estimación:	**Valor:**	**Dependencias:**
Descripción		
Como [operario], quiero [recibir notificación sobre nuevas OF emitidas], para [iniciar el proceso de fabricación, en cuanto termine el proceso en curso].		
Criterios de aceptación		
• El sistema debe mostrar un mensaje indicando las órdenes de fabricación nuevas programadas para iniciar el día actual. • El mensaje debe permanecer en la pantalla del punto de control por un tiempo definido (15 segundos) y luego debe ocultarse. • Las ordenes que ya se mostraron en el mensaje, deben ser marcadas por el sistema como notificadas.		

5.1.2.14 Consultar las órdenes de fabricación en proceso.

14	**Consultar las órdenes de fabricación en proceso.**	
Estimación:	**Valor:**	**Dependencias:**
Descripción		
Como [planificador de producción], quiero [consultar las órdenes de fabricación que están en proceso], para conocer [el avance de estas validar si el trabajo se está ejecutando de acuerdo al tiempo estimado].		
Criterios de aceptación		
• El usuario ingresa parámetros para consultar las ordenes de fabricación: fecha de emisión, estado de la OF, tipo de OF, etc. • El sistema debe mostrar las órdenes de fabricación de acuerdo a los filtros. • El sistema debe mostrar los datos tales como: OF, descripción, cliente, fecha de entrega, porcentaje de avance, hora de inicio y fin.		

5.1.2.15 Consultar avance de los procesos de una OF.

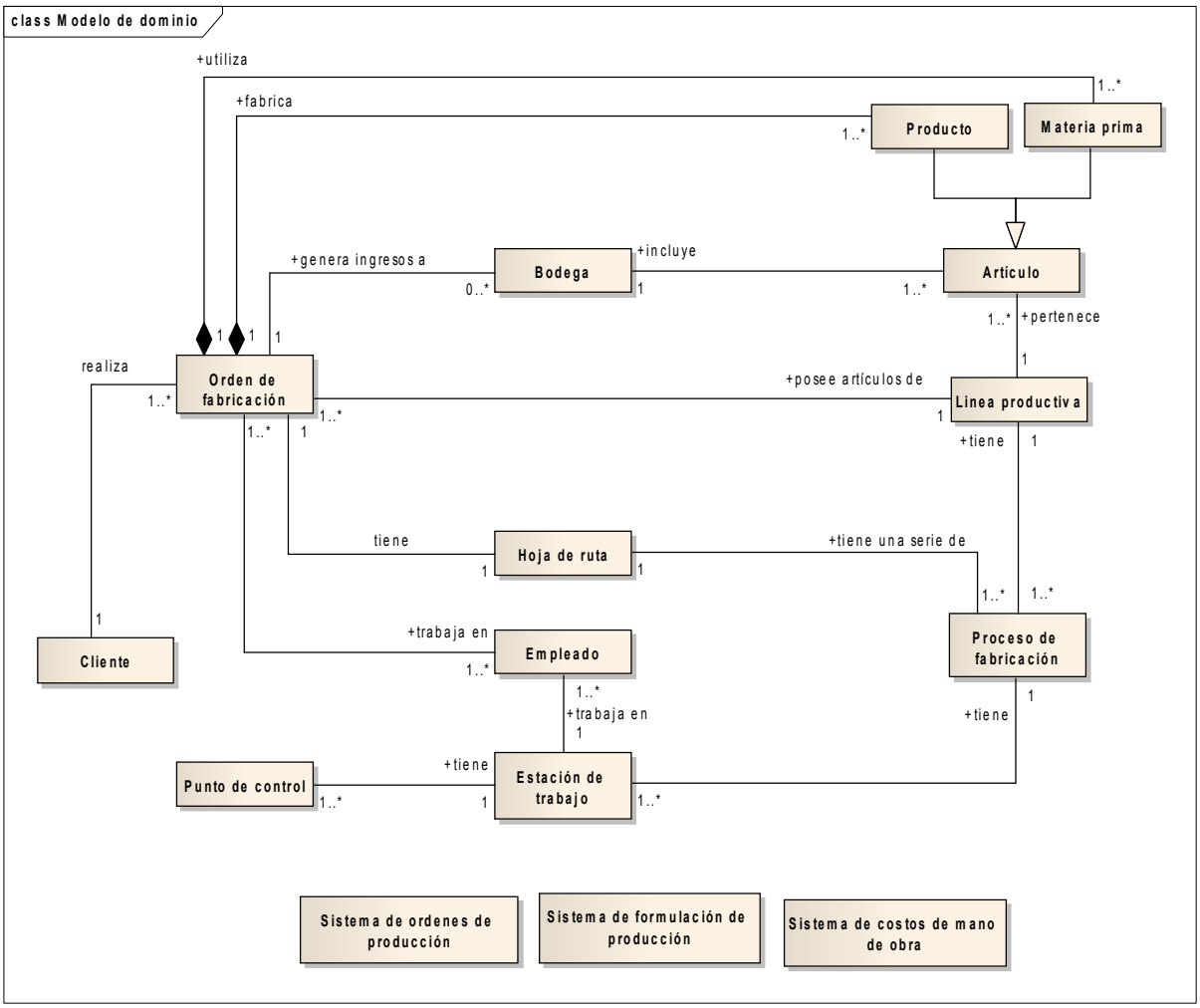
15	**Consultar avance de los procesos de una OF.**	
Estimación:	**Valor:**	**Dependencias:**
Descripción		
Como [planificador de producción], quiero [consultar el avance de la OF por proceso de fabricación], para verificar [el estado y avance de cada proceso].		
Criterios de aceptación		
• El sistema debe permitir el ingreso o selección de una OF, para revisar el avance de sus procesos. • El sistema debe mostrar el detalle de procesos, el estado actual, el porcentaje de avance, la hora de inicio y fin.		

5.2 Especificación de requerimientos de software

5.2.1 Modelo de dominio

Para diseñar una solución informática capaz de realizar tracking de procesos de manufactura, es necesario crear el modelo conceptual, identificando los conceptos y relaciones presentes entre cada concepto relacionado con el problema a resolver. Por tal razón, en la figura 3 se muestra el modelo de dominio.

Figura 3 Modelo de dominio para registrar tracking de procesos de manufactura.



Nota. Conceptos clave y sus principales relaciones. Elaboración propia a partir de la sección "Requerimientos de software".

Una orden de fabricación (OF), puede surgir por pedidos de clientes o para mantener productos en stock. La OF utiliza materia prima para fabricar los productos, por ello un artículo puede ser producto terminado o materia prima.

La OF debe poseer artículos de una única línea de producción, ya que la línea es la que contiene el detalle de procesos de fabricación y el orden en que se deben ejecutar. Esta asociación es la que permite generar la hoja de ruta que sus productos deben seguir.

Un proceso puede ser realizado en una o varias estaciones de trabajo, donde cada estación debe tener asignado uno o varios puntos de control para que los empleados que trabajan en determinada OF, marquen el inicio y fin de cada proceso.

5.2.2 Casos de uso

En esta sección, se describen los casos de uso que completan el comportamiento de las necesidades descritas en las historias de usuario.

5.2.2.1 C1 – Emitir OF

Objetivo:	Este caso de uso permite al usuario crear una orden de fabricación y colocarla en proceso, con lo cual se generará la hoja de ruta.
Prioridad:	Alta.
Actores:	Planificador de producción.
Pre-condiciones:	Tener acceso al sistema y a las funciones que el caso de uso describe.
Flujo normal:	1- El usuario selecciona la opción “Emitir orden de fabricación”. 2- El usuario ingresa los datos para crear la orden de fabricación: - Seleccionar el cliente que solicita el producto. - Ingresar la fecha de entrega, fecha estimada de producción, observación, etc. - Seleccionar los productos a fabricar. - La materia prima a utilizar puede ser ingresada manualmente, o esta puede ser estimada por un Sistema de formulación de producción. 3- El usuario selecciona la opción guardar, para registrar la orden de fabricación y esta queda en estado Emitida.
Flujos alternos:	3.1- Si por algún motivo ya no se trabajará la orden de fabricación, el usuario debe marcarla como "Anulada".
Post-condiciones:	El sistema genera la Hoja de ruta para fabricar la orden, y se muestra en listado de órdenes a producir.

5.2.2.2 C2 – Iniciar proceso

Objetivo:	Este caso de uso permite al usuario iniciar el proceso de fabricación de una orden.
Prioridad:	Alta.
Actores:	Operario.
Pre-condiciones:	Tener acceso al sistema y a las funciones que el caso de uso describe.
Flujo normal:	1- El usuario selecciona la opción “Iniciar”. 2- El usuario ingresa su código de empleado, por medio de teclado o escáner y el sistema valida que sea correcto. 3- El usuario ingresa el código de la orden de fabricación, por medio de teclado o escáner y el sistema valida que sea correcto. 4- El sistema valida que el punto de control pertenezca a la estación de trabajo que realiza el proceso de fabricación. 5- El sistema inicia la OF, y muestra el mensaje "OF iniciada correctamente!".
Flujos alternos:	2.1- Si el usuario no existe o no está activo, el sistema muestra el mensaje "Usuario no valido". 3.1- Si la orden de fabricación no existe el sistema muestra el mensaje "OF no existe." 3.2- Si la orden de fabricación no tiene el estado correcto, el sistema muestra el mensaje "OF debe estar Emitida o En proceso". 4.1- Si el punto de control no pertenece a la estación de trabajo donde se debe realizar el proceso, el sistema muestra el mensaje "Punto de control no válido para este proceso".
Post-condiciones:	El sistema verifica si hay un proceso iniciado y lo marca con "Pausa", e inicia el proceso actual y actualiza el estado de la orden de fabricación a "En proceso".

5.2.2.3 C3 – Pausar proceso

Objetivo:	Este caso de uso permite al usuario pausar el proceso de fabricación de una orden.
Prioridad:	Alta.
Actores:	Operario.
Pre-condiciones:	Tener acceso al sistema y a las funciones que el caso de uso describe.
Flujo normal:	1- El usuario selecciona la opción “Pausar”. 2- El usuario ingresa su código de empleado, por medio de teclado o escáner y el sistema valida que sea correcto. 3- El sistema busca el proceso iniciado y permite realizar la pausa.
Flujos alternos:	2.1- Si el usuario no existe o no está activo, el sistema muestra el mensaje "Usuario no valido". 3.1. Si no hay procesos iniciados, el sistema muestra el siguiente mensaje: "No hay procesos iniciados". 3.2. Si el punto de control no pertenece a la estación de trabajo donde se debe realizar el proceso, el sistema muestra el mensaje "Punto de control no válido para este proceso".
Post-condiciones:	El sistema marca el proceso como "Pausado", para el empleado.

5.2.2.4 C4 – Terminar proceso

Objetivo:	Este caso de uso permite al usuario terminar el proceso de fabricación, y terminar la orden si el proceso a terminar es el último y los ingresos a bodega ya están completos.
Prioridad:	Alta.
Actores:	Operario.
Pre-condiciones:	Tener acceso al sistema y a las funciones que el caso de uso describe.
Flujo normal:	1- El usuario selecciona la opción “Terminar”. 2- El usuario ingresa su código de empleado, por medio de teclado o escáner y el sistema valida que sea correcto. 3- El usuario ingresa el código de la orden de fabricación, por medio de teclado o escáner y el sistema valida que sea correcto. 4- El sistema termina el proceso, y muestra el mensaje "Proceso terminado correctamente!".
Flujos alternos:	2.1- Si el usuario no existe o no está activo, el sistema muestra el mensaje "Usuario no valido". 3.1- Si la orden de fabricación no existe el sistema muestra el mensaje "OF no existe." 3.2- Si la orden de fabricación no tiene el estado correcto, el sistema muestra el mensaje "OF debe estar Emitida o En proceso". 3.3- Si el punto de control no pertenece a la estación de trabajo donde se debe realizar el proceso, el sistema muestra el mensaje "Punto de control no válido para este proceso". 4.1. Si es el último proceso y ya se realizaron todos los ingresos a bodega, el sistema Termina la OF.
Post-condiciones:	El sistema marca el proceso como "Terminado". Además marca la orden como "Terminada", si ya se realizaron todos los ingresos a bodega.

5.2.2.5 C5 – Ingresar producto a bodega

Objetivo:	Este caso de uso permite al usuario realizar los ingresos del producto terminado a bodega.
Prioridad:	Alta.
Actores:	Operario.
Pre-condiciones:	Tener acceso al sistema y a las funciones que el caso de uso describe.
Flujo normal:	1- El usuario selecciona la opción “Ingresar a bodega”. 2- El usuario ingresa su código de empleado, por medio de teclado o escáner y el sistema valida que sea correcto. 3- El usuario ingresa el código de la orden de fabricación, por medio de teclado o escáner y el sistema valida que sea correcto. 4- El usuario ingresa el código del producto, por medio de teclado o escáner y el sistema valida que sea correcto. 5- El usuario ingresa la cantidad del producto, por medio de teclado, y el sistema valida que la cantidad no exceda de la detallada en la orden. 6- El sistema permite terminar la orden de fabricación, si ya se realizaron todos los ingresos.
Flujos alternos:	2.1- Si el usuario no existe o no está activo, el sistema muestra el mensaje "Usuario no valido". 3.1- Si la orden de fabricación no existe el sistema muestra el mensaje "OF no existe." 3.2- Si la orden de fabricación no tiene el estado correcto, el sistema muestra el mensaje "OF debe estar Emitida o En proceso". 3.3- Si el punto de control no pertenece a la estación de trabajo donde se debe realizar el proceso, el sistema muestra el mensaje "Punto de control no válido para este proceso". 4.1. Si el producto no está en el detalle de la OF, el sistema muestra el mensaje "Producto no válido". 5.1. Si la cantidad sobrepasa la detallada en la OF, el sistema muestra el mensaje "Cantidad no válida". 6.1. Si hay ingresos pendientes, el sistema muestra el mensaje "OF tiene ingresos pendientes".
Post-condiciones:	El sistema registra los ingresos y marca la orden como "Terminada", si dichos ingresos ya están completos.

5.2.2.6 C6 – Notificar nueva OF.

Objetivo:	Notificar al usuario sobre nuevas OF, pendientes de iniciar.
Prioridad:	Alta.
Actores:	Operativo.
Pre-condiciones:	Tener acceso al sistema y a las funciones que el caso de uso describe.
Flujo normal:	1- El sistema verifica si hay nuevas órdenes de fabricación pendientes de notificar para un proceso de fabricación. 2- Si hay OFs pendientes de notificar, el sistema muestra el mensaje: "Nuevas OFs: 190, 191,192". 3- El mensaje debe permanecer visible por un tiempo definido (15 segundos) y luego debe ocultarse.
Flujos alternos:	2.1- Si no hay OFs pendientes, no se debe mostrar ningún mensaje.
Post-condiciones:	El sistema debe marcar las OFs como notificadas.

5.2.2.7 C7 – Consultar ordenes de fabricación en proceso

Objetivo:	Permite al usuario consultar las órdenes en proceso y el avance en la producción.
Prioridad:	Alta.
Actores:	Planificador de producción.
Pre-condiciones:	Tener acceso al sistema y a las funciones que el caso de uso describe.
Flujo normal:	1- El usuario selecciona la opción “Ordenes de fabricación en proceso”. 2- Inicialmente el sistema muestra las ordenes de fabricación emitida en los últimos 15 días (valor que puede ser configurado). - Numero de OF. - Descripción. - Cliente. - Estado. - % avance. - Fecha de entrega. - Fecha y hora de inicio. - Fecha y hora de fin. 3- El usuario puede filtrar las ordenes por fecha emisión, tipo, cliente, estado de la OF y Proceso de fabricación. 4- El sistema muestra la opción "Salir". 5- El usuario selecciona la opción "Salir".
Flujos alternos:	2.1- El usuario puede cambiar los filtros y el sistema debe mostrar la información tal como se indica en el paso 2 del flujo normal de eventos. 4.1- El sistema muestra adicionalmente las opciones para ver detalle de productos y detalle de procesos de la orden.
Post-condiciones:	Las órdenes de fabricación han sido consultadas por el usuario.

5.2.2.8 C8 – Consultar avance de los procesos de una OF.

Objetivo:	Permite al usuario consultar los procesos de fabricación y el avance de estos para una orden específica.
Prioridad:	Alta.
Actores:	Planificador de producción.
Pre-condiciones:	Tener acceso al sistema y a las funciones que el caso de uso describe.
Flujo normal:	1- El usuario selecciona la opción “Avance de los procesos de una OF”. 2- El usuario ingresa o selecciona la orden de fabricación a consultar. 3- El sistema muestra la información de acuerdo a los filtros: - Proceso - Estado. - Inicio. - Fin. - Cantidad solicitada. - Cantidad producida. - % avance. - 5- El sistema muestra la opción "Salir". 6- El usuario selecciona la opción "Salir".
Flujos alternos:	2.1- El usuario puede cambiar los filtros y el sistema debe mostrar la información tal como se indica en el paso 2 del flujo normal de eventos.
Post-condiciones:	Los procesos de una orden de fabricación han sido consultados por el usuario.

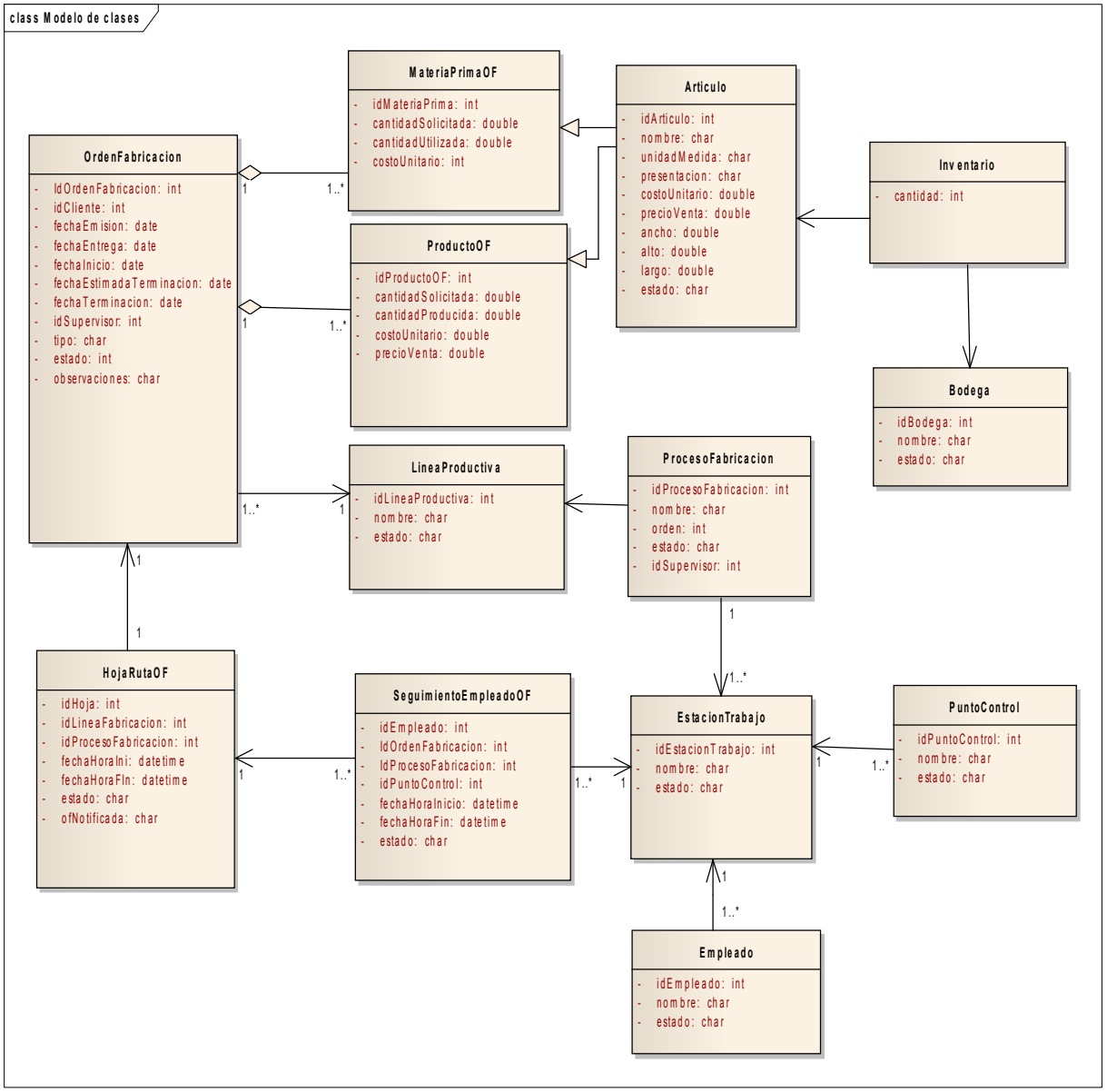
5.2.3 Diagrama de clases

5.2.3.1 Modelo de clases

Las clases que conformarán la aplicación para registrar el tracking de los procesos de manufactura, se muestran en la figura 4, estas se identifican a partir del modelo de dominio. El modelo de dominio muestra la relación de los conceptos relevantes dentro de la aplicación; mientras el diagrama de clase muestra también los atributos que tiene cada clase.

El diagrama de la figura 4 muestra las clases correspondientes a las clases de tipo entidad, es decir, las que finalmente guardarán información en la base de datos.

Figura 4 Modelo de clases para registrar tracking de procesos de manufactura.

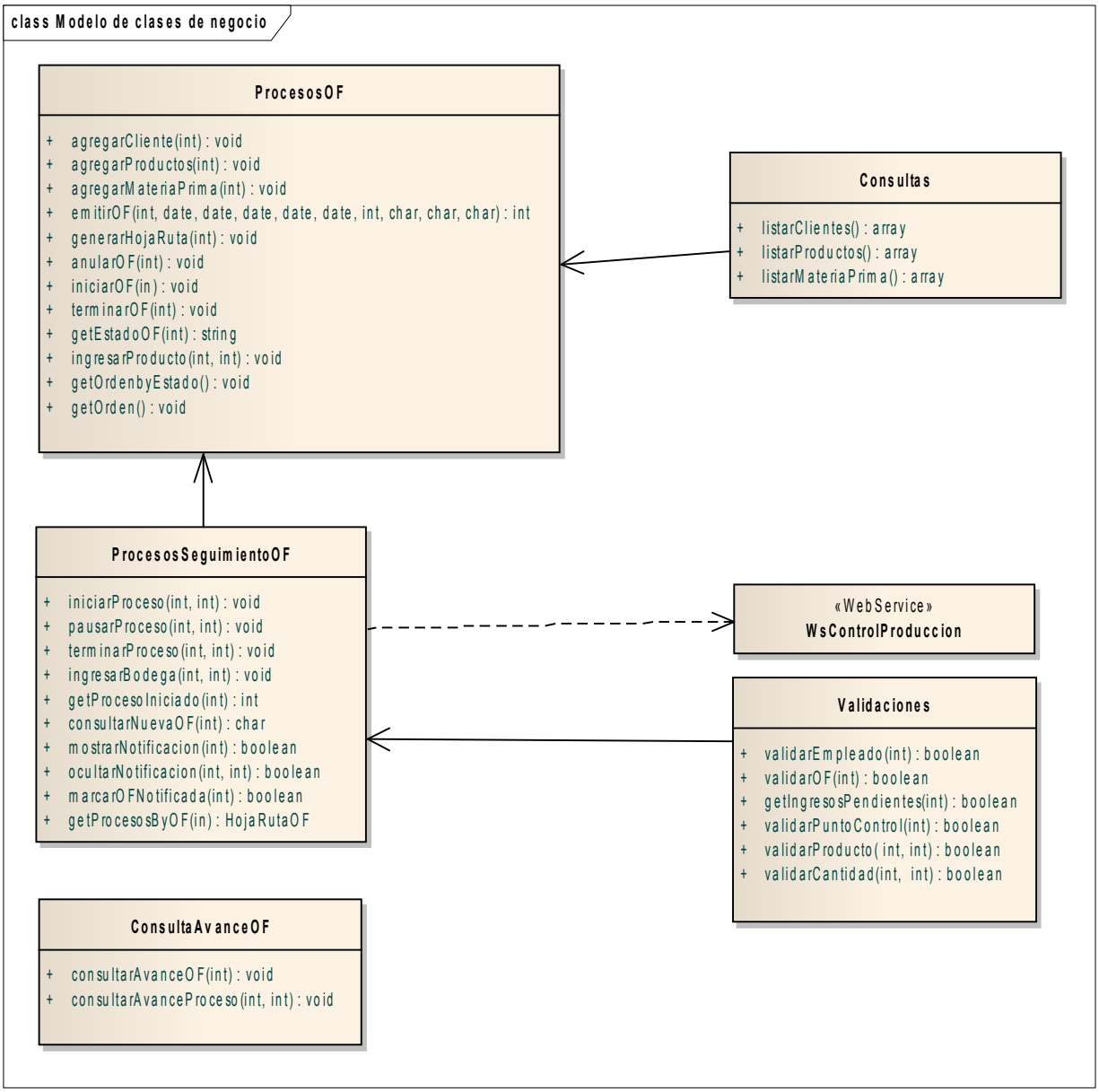


Nota. Clases principales, relación y cardinalidad entre ellas. Elaboración propia a partir de la sección "Requerimientos de software".

5.2.3.2 Modelo de clases de negocio.

En la sección anterior, se identificó las clases base de la aplicación, mientras que en esta sección, se presentan las clases de negocio que comprenden los métodos y validaciones que se deben realizar para registrar el tracking de procesos de manufactura. Es importante recordar que algunos de estos procesos serán utilizados desde el prototipo de hardware propuesto.

Figura 5 Modelo de clases de negocio para registrar tracking de procesos de manufactura.



Nota. Clases de negocio. Elaboración propia a partir de la sección "Requerimientos de software".

5.2.4 Diagrama de estado

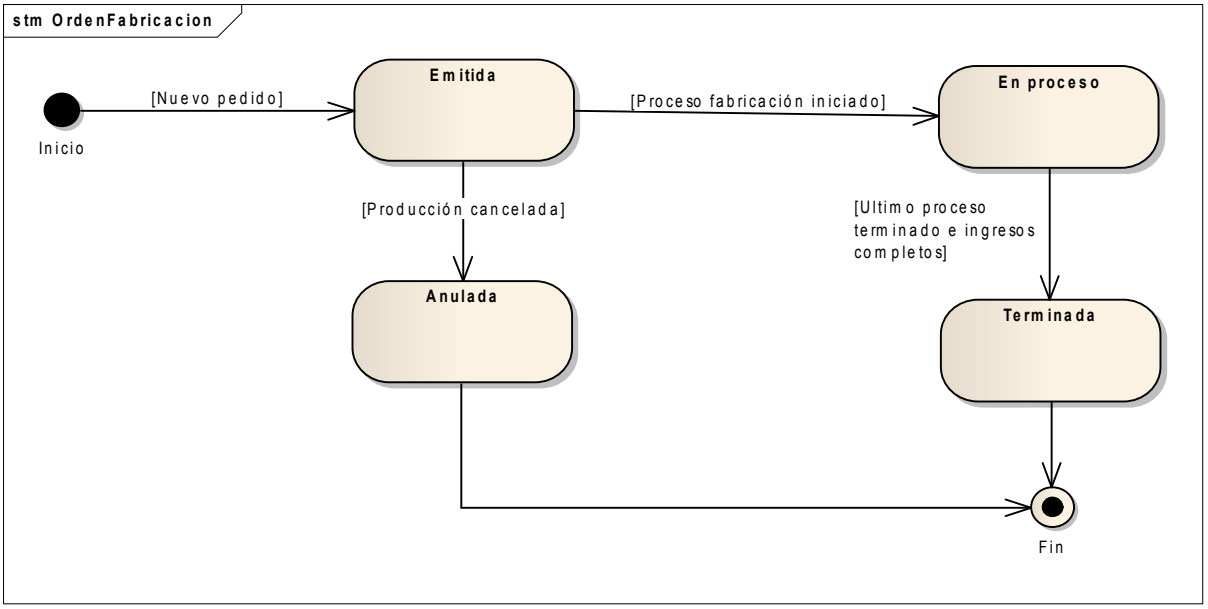
Antes de revisar a detalle la secuencia de cada proceso, es necesario conocer los estados que tendrá cada entidad.

5.2.4.1 Orden de fabricación

Una orden de fabricación, puede estar Emitida, En proceso, Terminada o Anulada.

- Emitida. Cuando la empresa recibe un nuevo pedido, el usuario crea la orden de fabricación y la marca como Emitida, generándose la hoja de ruta para que la planta de producción pueda iniciar el proceso de producción.
- En proceso. La orden se marca en este estado, cuando la planta de producción ya inició el primer proceso.
- Terminada. Es cuando la orden de fabricación ya está ingresada en bodega.
- Anulada. Es cuando el usuario cancela la producción de la OF.

Figura 6 Diagrama de estados de una orden de fabricación.



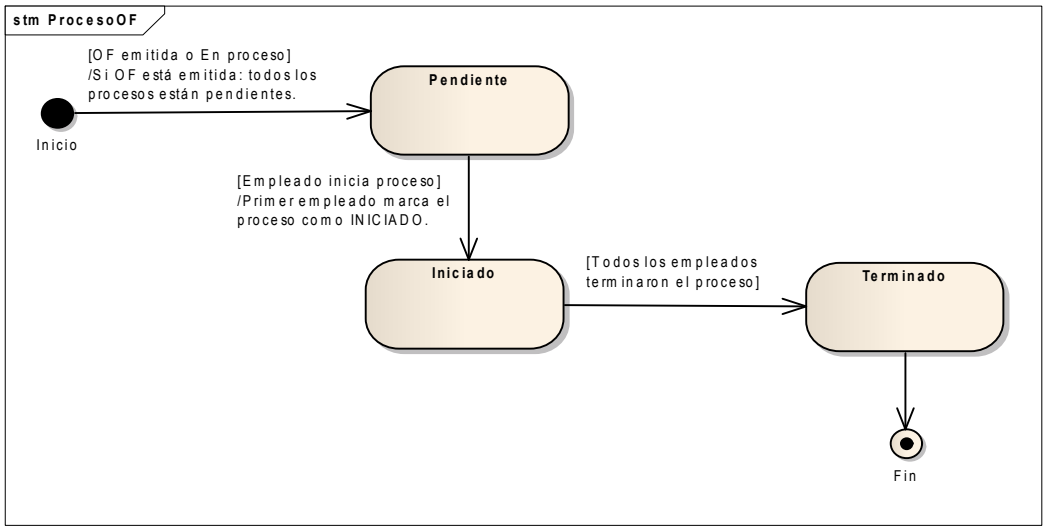
Nota. Estados de una orden de fabricación. Elaboración propia a partir de la sección "Requerimientos de software".

5.2.4.2 Proceso de fabricación

Los procesos de fabricación, pueden tener los estados: Pendiente, Iniciado y Terminado.

- Pendiente. Cuando la orden de fabricación está emitida o en proceso.
- Iniciado. Cuando el primer empleado inicia el proceso.
- Terminado. Cuando todos los empleados terminan el proceso.

Figura 7 Diagrama de estados de un proceso de fabricación.



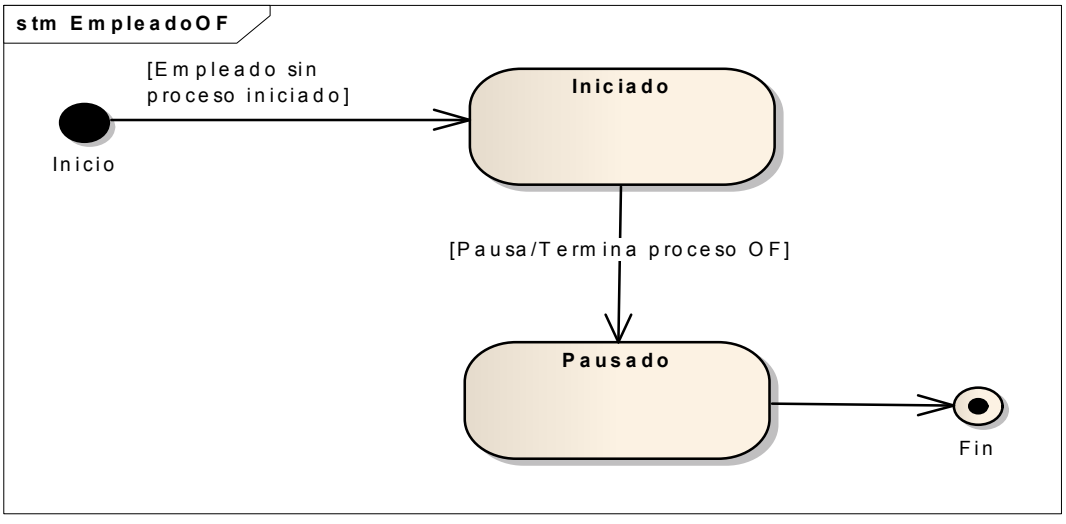
Nota. Estados de un proceso de fabricación. Elaboración propia a partir de la sección "Requerimientos de software".

5.2.4.3 Empleado trabajando en un proceso de fabricación

También, es necesario registrar una bitácora de los procesos en que un empleado está trabajando, en este caso los estados pueden estar en estado: Iniciado o Pausado.

- Iniciado. Empleado sin procesos iniciados.
- Pausado. Empleado en pausa o con proceso terminado.

Figura 8 Diagrama de estados de un proceso de fabricación.



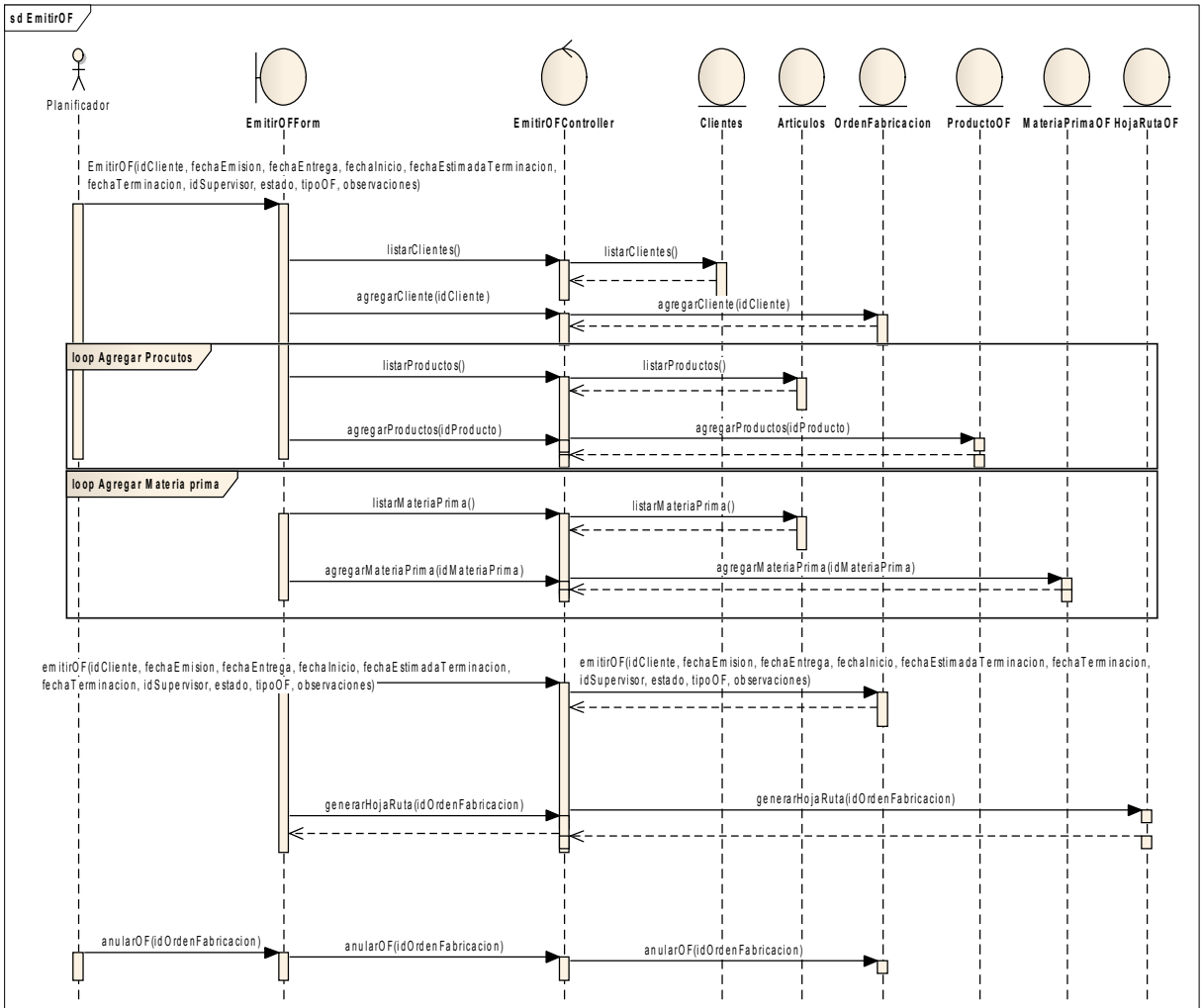
Nota. Estados de un empleado realizando un proceso de fabricación. Elaboración propia a partir de la sección "Requerimientos de software".

5.2.5 Diagrama de secuencia

En esta sección, se presentan los diagramas de secuencia correspondientes a los casos de uso descritos.

5.2.5.1 Emitir OF

Figura 9 Diagrama de secuencia para Emitir OF.



Emitir OF

Flujo normal:

1- El usuario selecciona la opción "Emitir orden de fabricación".

2- El usuario ingresa los datos para crear la orden de fabricación:

- Seleccionar el cliente que solicita el producto.
- Ingresar la fecha de entrega, fecha estimada de producción, observación, etc.
- Seleccionar los productos a fabricar.
- La materia prima a utilizar puede ser ingresada manualmente, o esta puede ser estimada por un Sistema de formulación de producción.

3- El usuario selecciona la opción guardar, para registrar la orden de fabricación y esta queda en estado Emitida.

Flujos alternos:

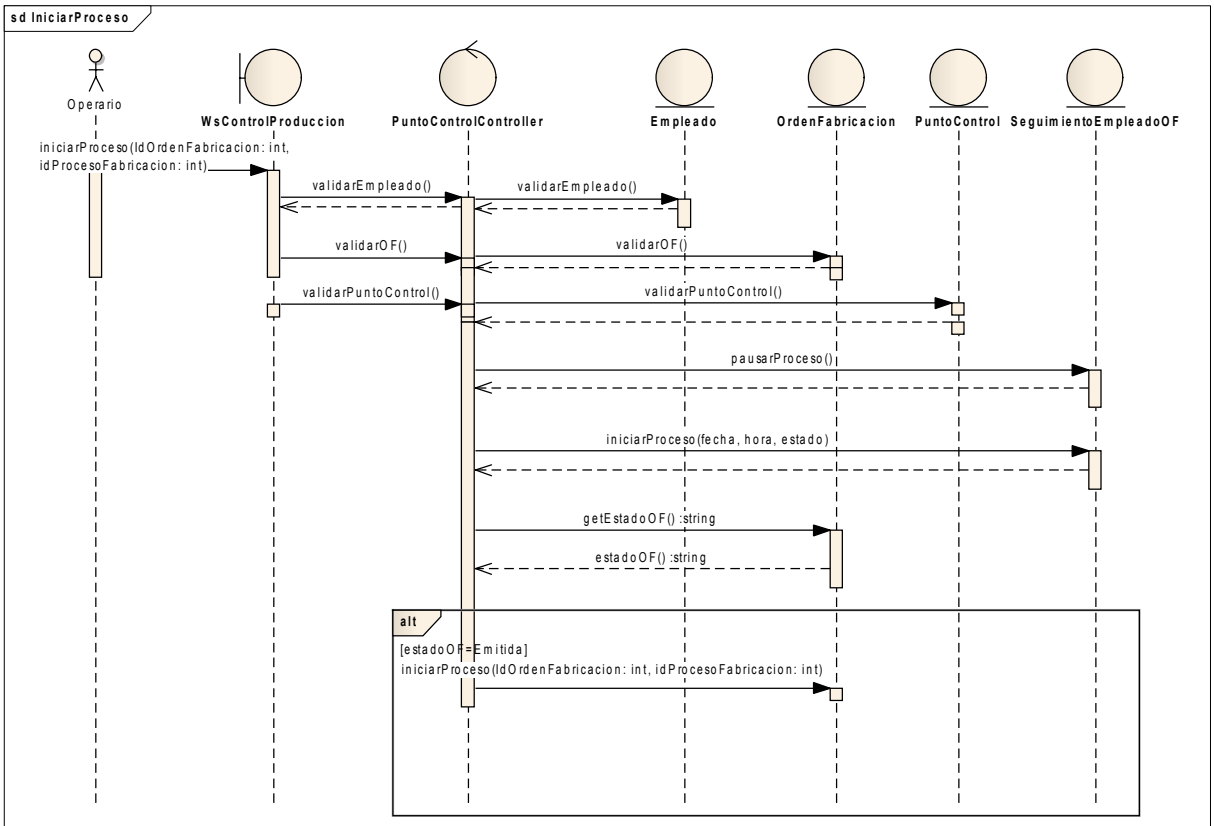
3.1- Si por algún motivo ya no se trabajará la orden de fabricación, el usuario debe marcarla como "Anulada".

Post-condiciones:

El sistema genera la Hoja de ruta para fabricar la orden, y se muestra en listado de órdenes a producir.

5.2.5.2 Iniciar proceso

Figura 10 Diagrama de secuencia para Iniciar proceso.



Iniciar proceso

Flujo normal:

1- El usuario selecciona la opción "Iniciar".

2- El usuario ingresa su código de empleado, por medio de teclado o escáner y el sistema valida que sea correcto.

3- El usuario ingresa el código de la orden de fabricación, por medio de teclado o escáner y el sistema valida que sea correcto.

4- El sistema valida que el punto de control pertenezca a la estación de trabajo que realiza el proceso de fabricación.

5- El sistema inicia la OF, y muestra el mensaje "OF iniciada correctamente!".

Flujos alternos:

2.1- Si el usuario no existe o no está activo, el sistema muestra el mensaje "Usuario no valido".

3.1- Si la orden de fabricación no existe el sistema muestra el mensaje "OF no existe."
3.2- Si la orden de fabricación no tiene el estado correcto, el sistema muestra el mensaje "OF debe estar Emitida o En proceso".

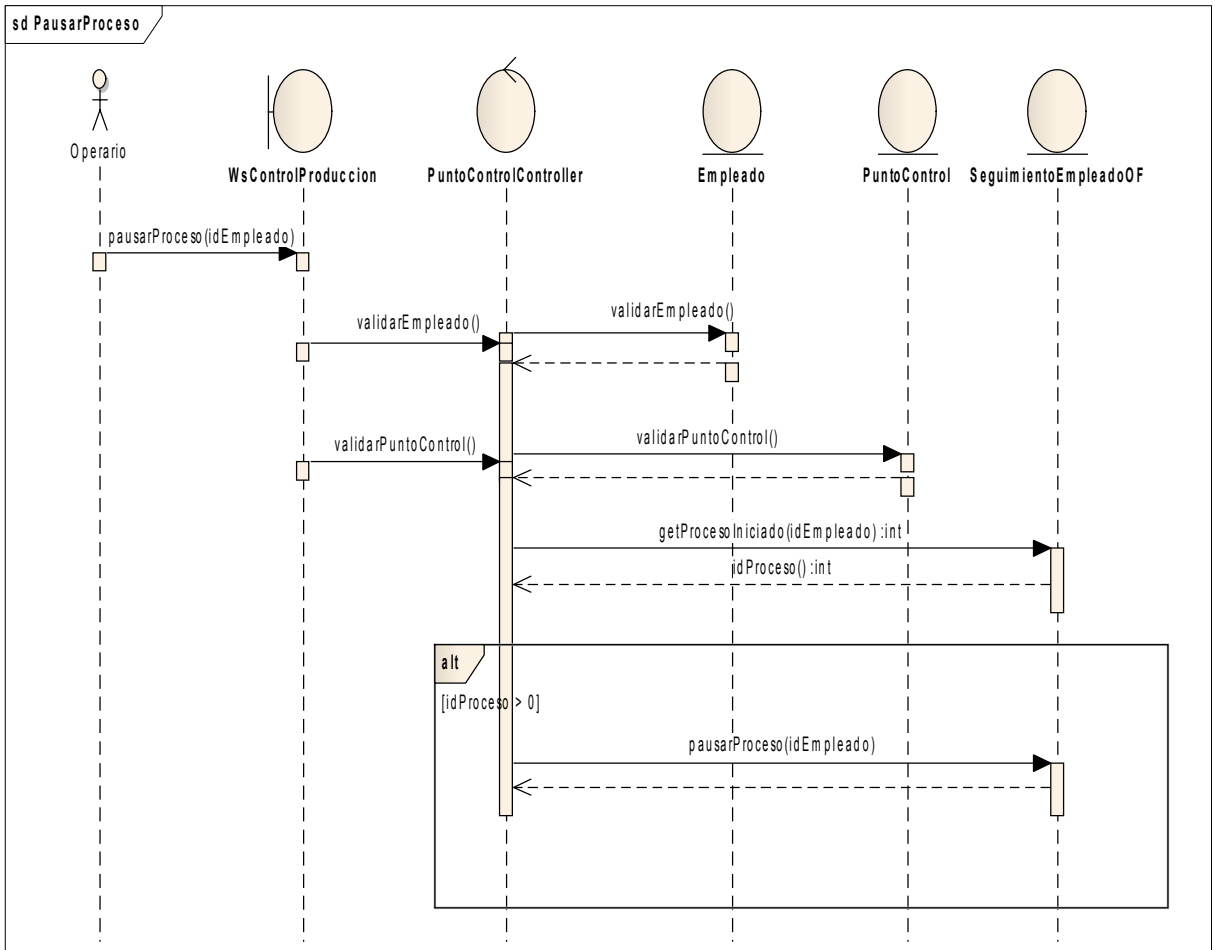
4.1- Si el punto de control no pertenece a la estación de trabajo donde se debe realizar el proceso, el sistema muestra el mensaje "Punto de control no válido para este proceso".

Post-condiciones:

El sistema verifica si hay un proceso iniciado y lo marca con "Pausa", e inicia el proceso actual y actualiza el estado de la orden de fabricación a "En proceso".

5.2.5.3 Pausar proceso

Figura 11 Diagrama de secuencia para pausar proceso



Pausar proceso

Flujo normal:

1- El usuario selecciona la opción "Pausar".

2- El usuario ingresa su código de empleado, por medio de teclado o escáner y el sistema valida que sea correcto.

3- El sistema busca el proceso iniciado y permite realizar la pausa.

Flujos alternos:

2.1- Si el usuario no existe o no está activo, el sistema muestra el mensaje "Usuario no valido".

3.1. Si no hay procesos iniciados, el sistema muestra el siguiente mensaje: "No hay procesos iniciados".

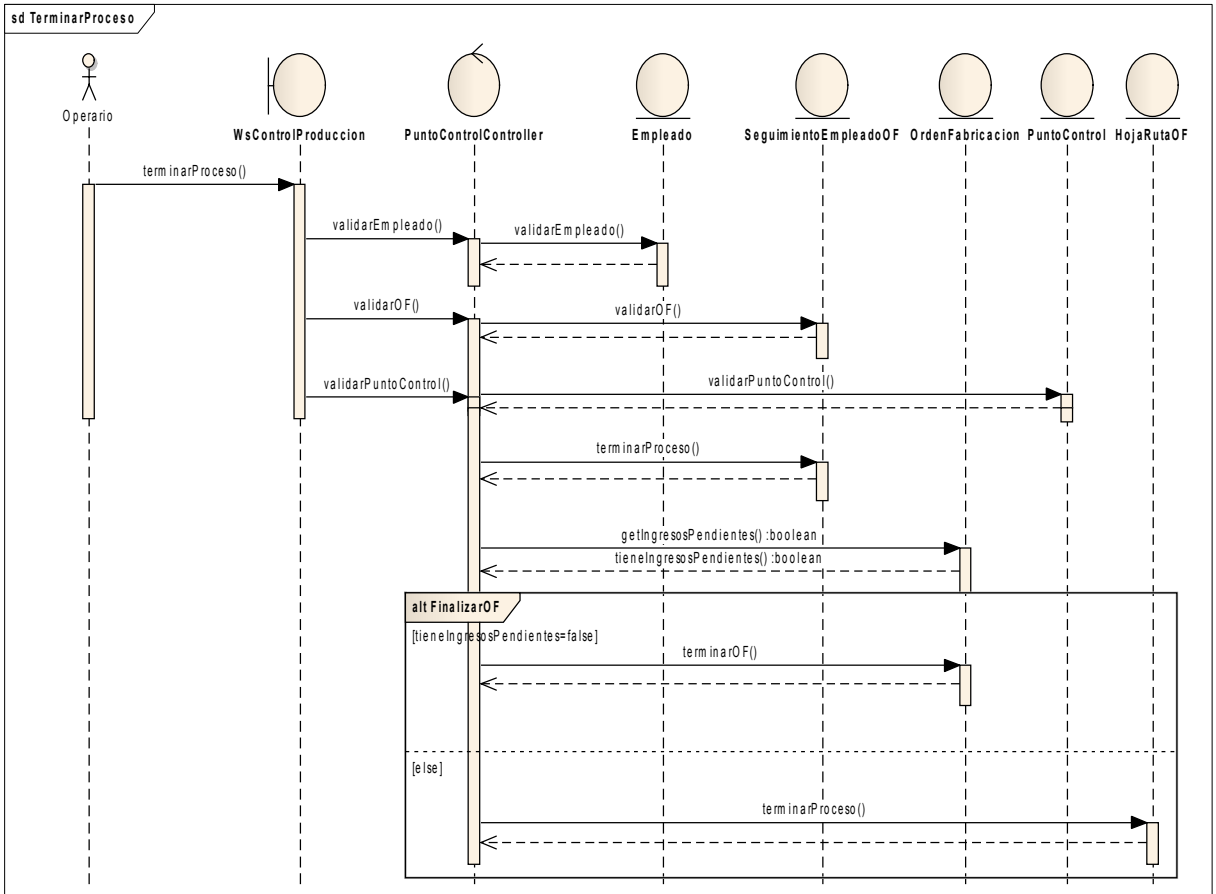
3.2. Si el punto de control no pertenece a la estación de trabajo donde se debe realizar el proceso, el sistema muestra el mensaje "Punto de control no válido para este proceso".

Post-condiciones:

El sistema marca el proceso como "Pausado", para el empleado.

5.2.5.4 Terminar proceso

Figura 12 Diagrama de secuencia para terminar proceso.



Terminar proceso

Flujo normal:

1- El usuario selecciona la opción "Terminar".

2- El usuario ingresa su código de empleado, por medio de teclado o escáner y el sistema valida que sea correcto.

3- El usuario ingresa el código de la orden de fabricación, por medio de teclado o escáner y el sistema valida que sea correcto.

4- El sistema termina el proceso, y muestra el mensaje "Proceso terminado correctamente!".

Flujos alternos:

2.1- Si el usuario no existe o no está activo, el sistema muestra el mensaje "Usuario no valido".

3.1- Si la orden de fabricación no existe el sistema muestra el mensaje "OF no existe."

3.2- Si la orden de fabricación no tiene el estado correcto, el sistema muestra el mensaje "OF debe estar Emitida o En proceso".

3.3- Si el punto de control no pertenece a la estación de trabajo donde se debe realizar el proceso, el sistema muestra el mensaje "Punto de control no válido para este proceso".

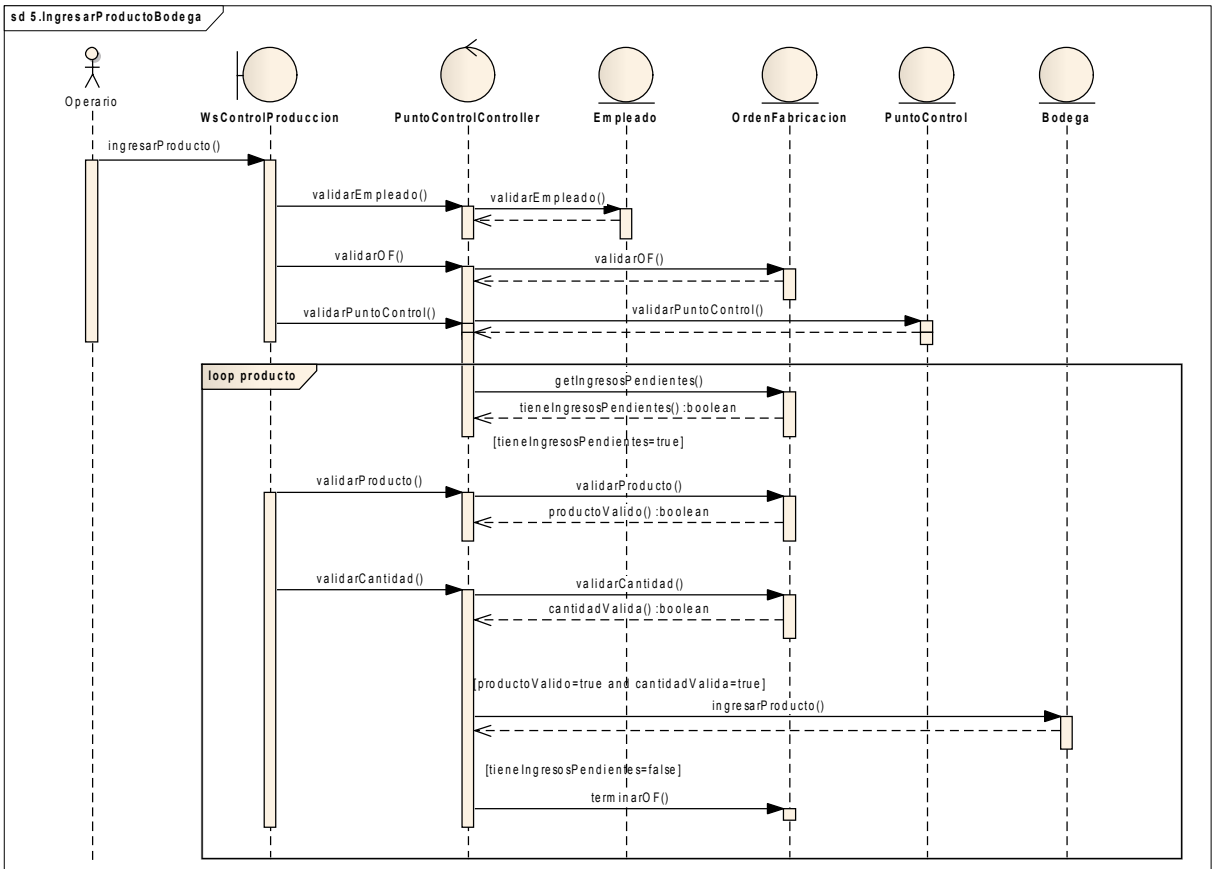
4.1. Si es el último proceso y ya se realizaron todos los ingresos a bodega, el sistema Termina la OF.

Post-condiciones:

El sistema marca el proceso como "Terminado". Además marca la orden como "Terminada", si ya se realizaron todos los ingresos a bodega.

5.2.5.5 Ingresar producto a bodega

Figura 13 Diagrama de secuencia para ingresar producto a bodega.



Ingresar producto a bodega

Flujo normal

1- El usuario selecciona la opción "Ingresar a bodega".

2- El usuario ingresa su código de empleado, por medio de teclado o escáner y el sistema valida que sea correcto.

3- El usuario ingresa el código de la orden de fabricación, por medio de teclado o escáner y el sistema valida que sea correcto.

4- El usuario ingresa el código del producto, por medio de teclado o escáner y el sistema valida que sea correcto.

5- El usuario ingresa la cantidad del producto, por medio de teclado, y el sistema valida que la cantidad no exceda de la detallada en la orden.

6- El sistema permite terminar la orden de fabricación, si ya se realizaron todos los ingresos.

Flujos alternos:

2.1- Si el usuario no existe o no está activo, el sistema muestra el mensaje "Usuario no valido".

3.1- Si la orden de fabricación no existe el sistema muestra el mensaje "OF no existe."

3.2- Si la orden de fabricación no tiene el estado correcto, el sistema muestra el mensaje "OF debe estar Emitida o En proceso".

3.3- Si el punto de control no pertenece a la estación de trabajo donde se debe realizar el proceso, el sistema muestra el mensaje "Punto de control no válido para este proceso".

4.1. Si el producto no está en el detalle de la OF, el sistema muestra el mensaje "Producto no válido".

5.1. Si la cantidad sobrepasa la detallada en la OF, el sistema muestra el mensaje "Cantidad no válida".

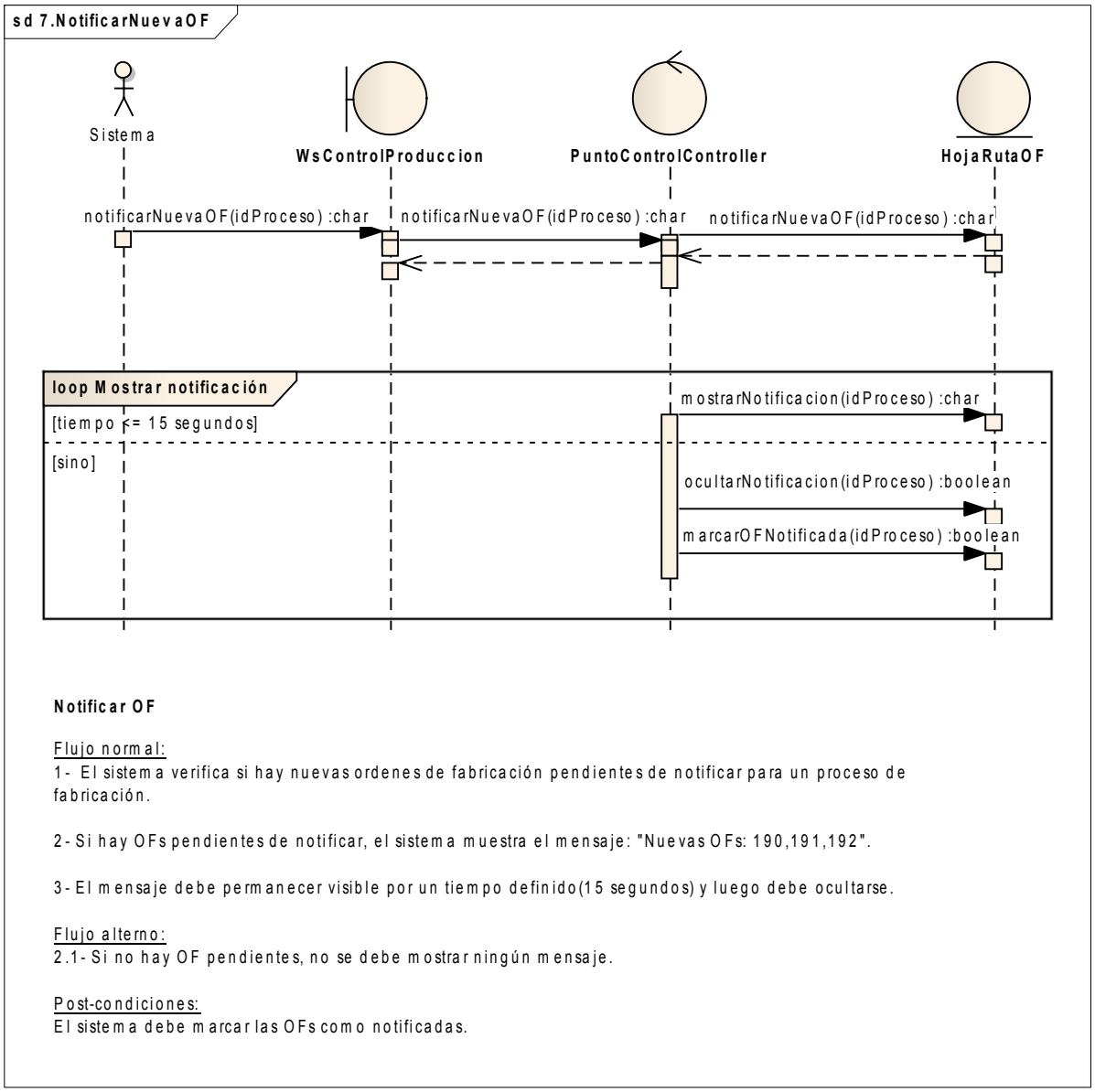
6.1. Si hay ingresos pendientes, el sistema muestra el mensaje "OF tiene ingresos pendientes".

Post-condiciones:

El sistema registra los ingresos y marca la orden como "Terminada", si dichos ingresos ya están completos.

5.2.5.6 Notificar nueva OF

Figura 14 Diagrama de secuencia para notificar nueva OF.



Notificar OF

Flujo normal:
1- El sistema verifica si hay nuevas ordenes de fabricación pendientes de notificar para un proceso de fabricación.

2- Si hay OFs pendientes de notificar, el sistema muestra el mensaje: "Nuevas OFs: 190,191,192".

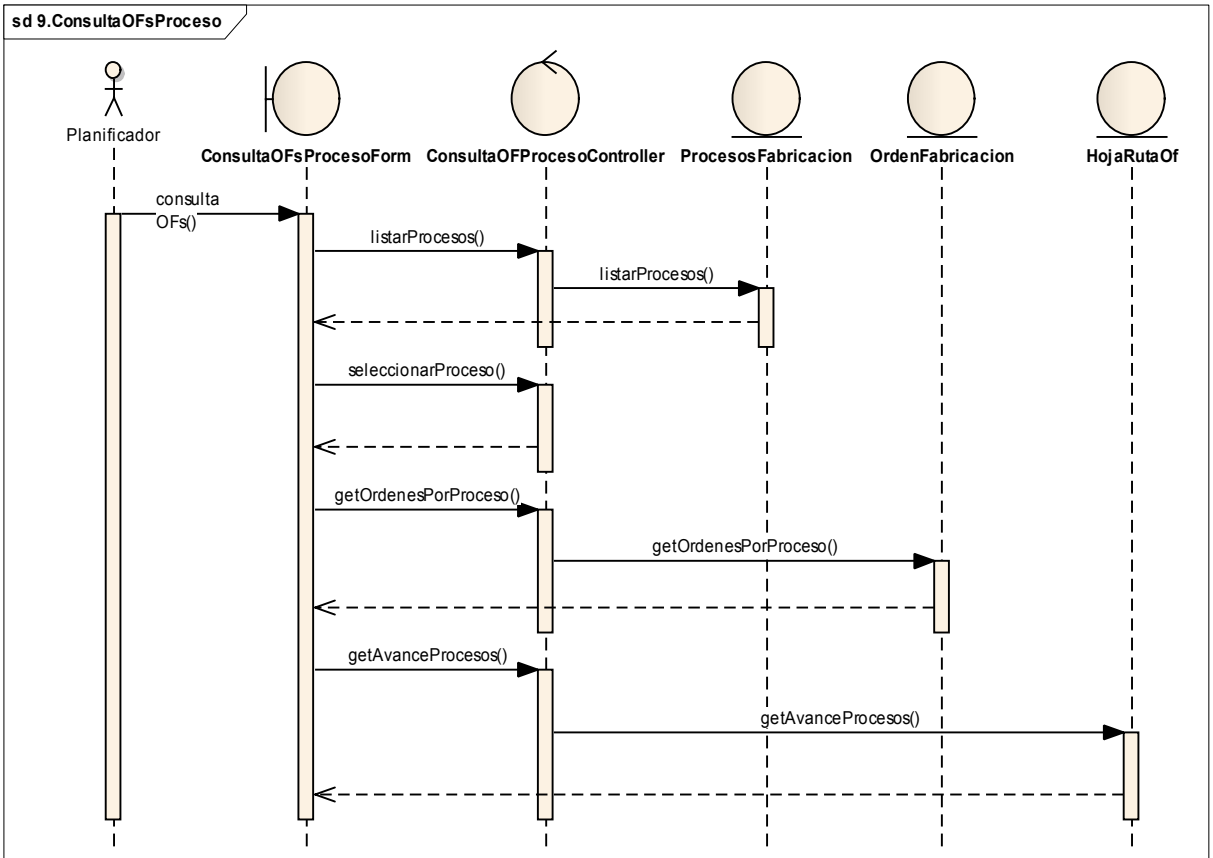
3- El mensaje debe permanecer visible por un tiempo definido(15 segundos) y luego debe ocultarse.

Flujo alterno:
2.1- Si no hay OF pendientes, no se debe mostrar ningún mensaje.

Post-condiciones:
El sistema debe marcar las OFs como notificadas.

5.2.5.7 Consultar Órdenes de Fabricación en Proceso

Figura 15 Diagrama de Secuencia para Consultar Avances Órdenes de Fabricación en Proceso



1- El usuario selecciona la opción "Ordenes de fabricación en proceso".

2- Inicialmente el sistema muestra las ordenes de fabricación emitida en los últimos 15 días (valor que puede ser configurado).

- Numero de OF.
- Descripción.
- Cliente.
- Estado.
- % avance.
- Fecha de entrega.
- Fecha y hora de inicio.
- Fecha y hora de fin.

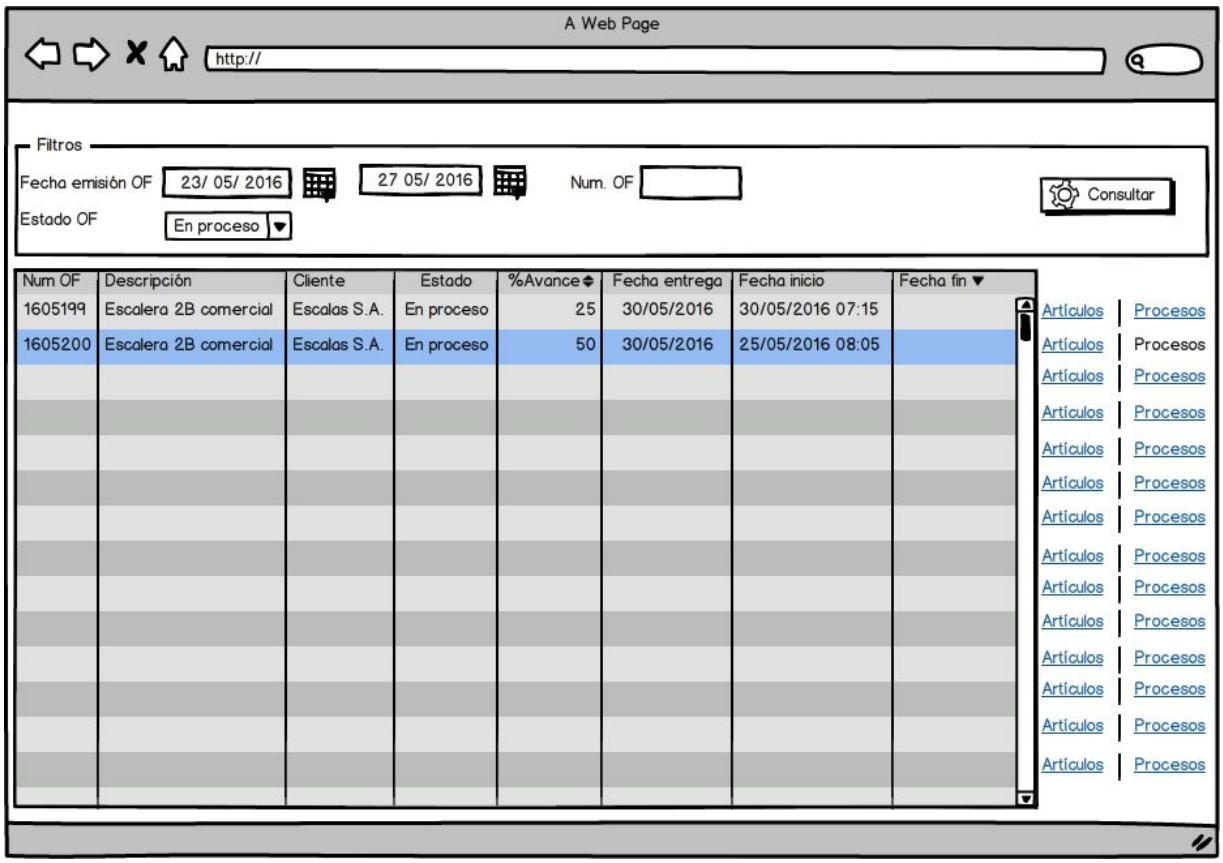
3- El usuario puede filtrar las ordenes por fecha emisión, tipo, cliente, estado de la OF y Proceso de fabricación.

5.2.6 Interfaz de usuario

En esta sección se presenta el diseño de las principales pantallas que servirán para consultar el estado y avance de las ordenes de fabricación y sus procesos.

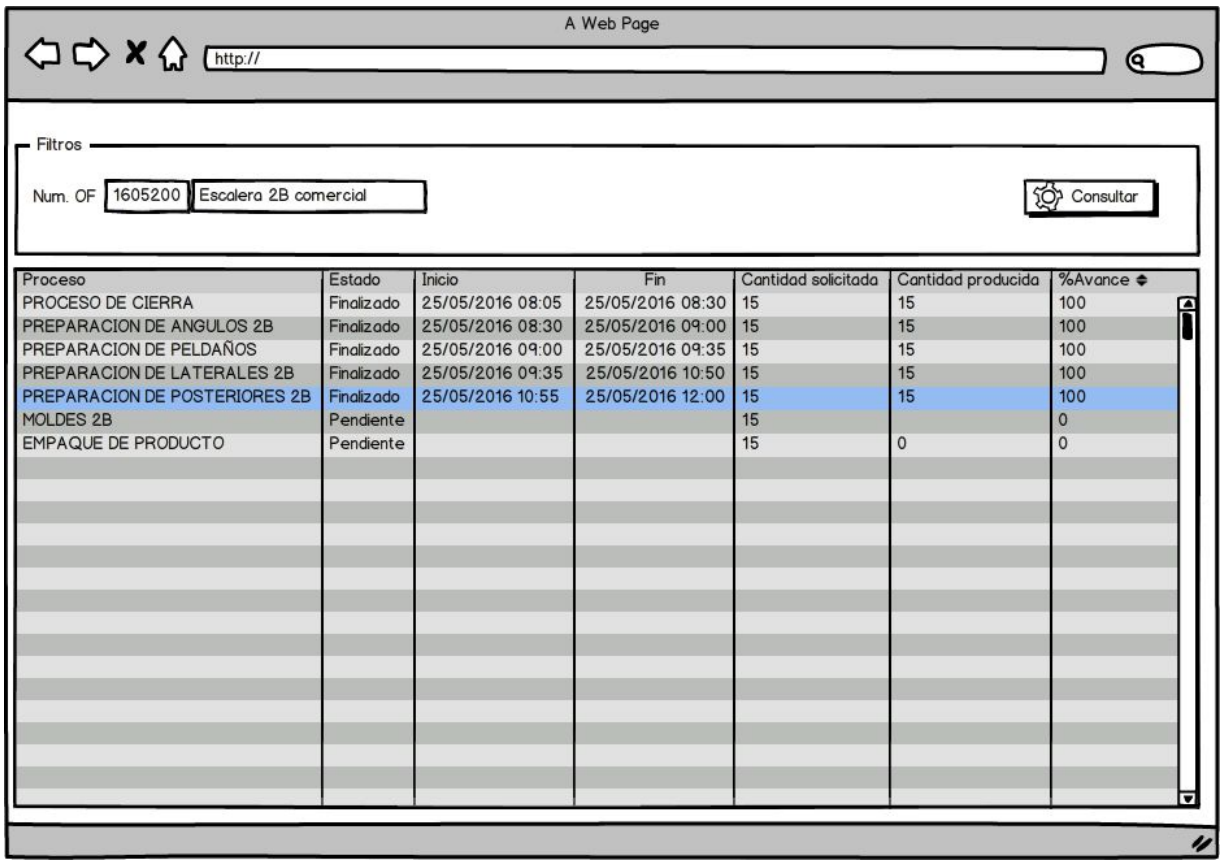
En la figura16 se muestra la pantalla para consultar las ordenes de fabricación, la información puede ser filtrada por fecha de emisión, estado y número de orden; la opción "Consultar", mostrará las ordenes, el porcentaje de avance, la fecha y hora de inicio y finalización. Para visualizar el detalle de artículos de cada orden, se utilizará la opción "Artículos" y la opción "Procesos", mostrará el detalle de procesos de fabricación y su avance.

Figura 16 Consulta de ordenes de fabricación.



El detalle de procesos de fabricación y el avance de cada uno, se consultará por medio de una pantalla similar a la que se presenta en la figura 17.

Figura 17 Consulta de procesos de fabricación de una orden específica.

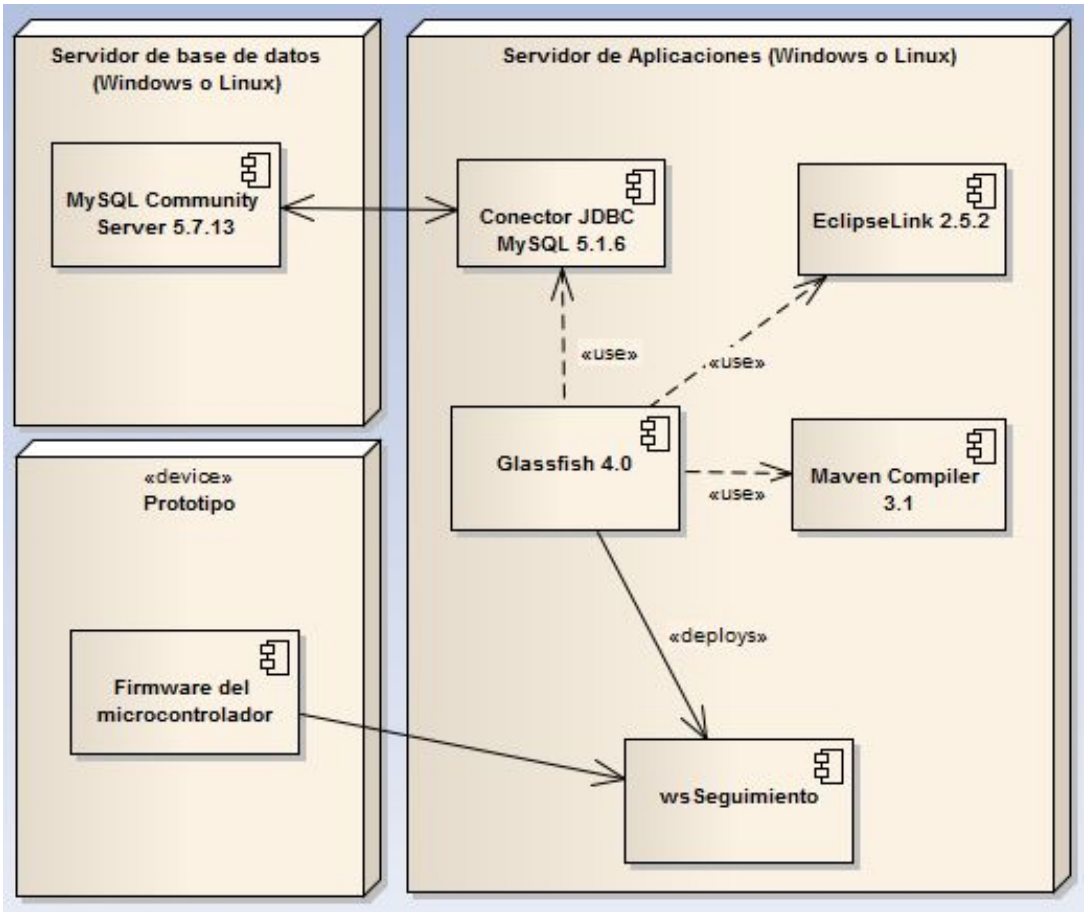


5.2.7 Diagrama de despliegue

La API propuesta es una aplicación web que almacena la información en una base de datos MySQL, desarrollada en lenguaje Java y desplegada en el servidor de aplicaciones Glassfish 4.0. La aplicación puede desplegarse en cualquier sistema operativo que soporte MySQL y Glassfish.

El prototipo que sirve como punto de control, utilizará un firmware con el cual se capturarán los datos en el piso de producción, y se comunicará con la API por medio de Web Services. En la Figura 18, se describe la forma en que se comunican e interactúan todos los componentes involucrados. La elección de la placa de desarrollo no afecta el diseño de la API, ni el despliegue de la solución.

Figura 18 Diagrama de despliegue para el tracking de procesos de manufactura.



El servidor de base de datos, contiene:

- MySQL Community Server 5.7.13. Es un gestor de base de datos relacional Open Source.

El servidor de aplicaciones está compuesto por los siguientes elementos:

- Conector JDBC MySQL 5.1.6. Es el conector que permite establecer la comunicación entre la base de datos y la aplicación.
- EclipeLink 2.5.2. Es el motor para realizar la persistencia en la base de datos.
- Glassfish 4.0. Es el servidor de aplicaciones donde se despliega la API.
- Maven Compiler 3.1. Es el software para gestionar las librerías, dependencias, complementos y otros elementos que utilizará la aplicación desarrollada.
- wsSeguimiento. Es el proyecto web escrito en lenguaje Java que contiene la lógica de negocio y los servicios disponibles para consumir desde el prototipo.

6 Diseño del prototipo

En esta sección, se explicará el diseño del prototipo propuesto para cumplir con los requerimientos definidos a través de las historias de usuario.

6.1 Diseño de hardware

6.1.1 Requerimientos de hardware

En la sección de especificación de requerimientos de software se expone todos los requisitos que se deben cumplir con la API. Sin embargo, algunas historias de usuarios son específicas de la API, mientras que otras son realizadas por el prototipo.

Las historias de usuario que deben ser cumplidas por el prototipo son las que se refieren a acciones realizadas en el piso de producción por los operarios; quienes dan seguimiento a la producción en tiempo real. Estas historias son:

- Iniciar proceso.
- Pausar proceso.
- Terminar proceso.
- Ingresar producto a bodega.

Además, se ha determinado que el prototipo debe cumplir con las siguientes características de hardware para satisfacer los requerimientos anteriores:

- Contar con una pantalla donde se despliegue información de las órdenes de fabricación y solicitar los datos requeridos para capturar el avance.
- Para el input de datos se requiere que posea teclado y que sea capaz de leer códigos de barras de órdenes de fabricación, empleados, artículos, etc.
- Contar con conectividad a red LAN o WLAN, para transmitir los datos capturados hacia el sistema donde se procesarán.
- Si se realiza conexión inalámbrica debe soportar métodos de autenticación segura, como WPA-Personal o WPA-Enterprise, no se acepta autenticación WEP ni abierta.
- Tener un consumo de energía menor o igual al de una computadora portátil promedio. Es decir, 20V y 3.25 A.

6.1.2 Selección de placa microcontroladora

Para la elección de la placa microcontroladora a utilizar en el prototipo se consideró cuatro alternativas. Cuando se habla de opciones para prototipado para la computación física y/o en la nube, las opciones con mejor posicionamiento en el mercado son: Arduino, Raspberry Pi y Beaglebone Black (14).

También se puede tomar como referencia empresas dedicadas a la venta y manufactura de componentes de hardware, electrónicos y componentes, como lo son Adafruit Industries y Sparkfun, donde se puede encontrar una cuarta opción, que es el Photon de Particle, entre otras.

La elección de una plataforma particular puede ser una tarea difícil, ya que depende de criterios como el precio, el desempeño, y las capacidades que son apropiadas a lo que se trata de alcanzar. A pesar de la alternativa seleccionada en este trabajo viene de criterios relativamente objetivos, alguien más podría elegir una solución completamente diferente para resolver el mismo problema de manera correcta (14) .

Después de la evaluación realizada, según lo descrito en la metodología, se determinó que la placa más adecuada para el prototipo es el Arduino Yun. En el Anexo 1, llamado "Evaluación de tarjetas microcontroladoras", se pueden observar los criterios tomados en cuenta y la calificación ponderada en cada uno, cada tarjeta cuenta con una ficha técnica que describe sus capacidades. Los siguientes son algunos de los puntos a favor de esta tarjeta:

- Posee de fábrica la posibilidad de conectar el dispositivo a la red tanto vía Ethernet como por Wifi.
- Al conectar vía Wifi, puede utilizar conexión con seguridad empresarial WPA2-Enterprise, inclusive validarse con servidor RADIUS.
- Posee un puerto USB Host de fábrica para conectar dispositivos USB como teclado o escáner.
- El precio del prototipo completo, incluyendo pantalla LCD, teclado numérico, escáner y carcasa es favorable comparado con las otras opciones.
- Puede ser adquirido a nivel local.
- Gran cantidad de librerías y escudos disponibles para la ampliación de las funcionalidades.

Es necesario aclarar que Arduino Yun no es la placa más potente entre las alternativas evaluadas, y por ello se hará una mención a las características destacables de las otras tarjetas, así como sus desventajas.

El Raspberry Pi 2, que es la opción vigente al momento de la evaluación, es la placa con mayor capacidad entre las seleccionadas, básicamente es una computadora en un Chip, tiene varios puertos USB, un sistema Linux instalado y puede ser programado en varios lenguajes; sin embargo, es la placa con mayor consumo de corriente, y tiene menos capacidades en cuanto a computación física que las demás alternativas; además el prototipo tiende a ser de mayor tamaño.

El Photon, de Particle, es una opción potente, el chip es el más compacto de todos, la placa tiene el menor precio, conectividad Wifi y el respaldo de la nube de Particle; sin embargo, no tiene funcionalidad USB Host de fábrica (a la fecha en que se realizó la evaluación), lo que dificulta la conexión del escáner de código de barras. Además, dado que la solución está diseñada para funcionar a nivel empresarial, es una desventaja que no provea de momento de conectividad de tipo empresarial WPA2 Enterprise (a la fecha en que se realizó la evaluación).

El Beaglebone Black, es otro sistema en Chip, que cuenta con una distribución de Linux instalada y que tiene buenas prestaciones, soporta computación en tiempo real. Al igual que el Raspberry Pi, tiene un consumo relativamente alto de corriente, tiene una comunidad más reducida; y, en caso de desplegar la interfaz gráfica, se puede realizar únicamente con una pantalla HDMI, lo que encarecería el prototipo.

6.1.3 Diseño de hardware y conexiones

Las tareas que un operario podrá realizar desde el prototipo de hardware son básicamente: iniciar el trabajo en una orden de fabricación, definir pausas, indicar cuando ha terminado la tarea o proceso actual y realizar ingresos a bodega.

A partir de las acciones anteriores y tomando en cuenta la conveniencia del uso de códigos de barra, tanto para identificar la orden de fabricación, artículos producidos y empleados del taller de producción; se determina que el prototipo debe contar con los siguientes componentes:

Teclas para definir la acción a realizar y digitar números en caso de no contar con código de barra, o que este no pueda ser reconocido.

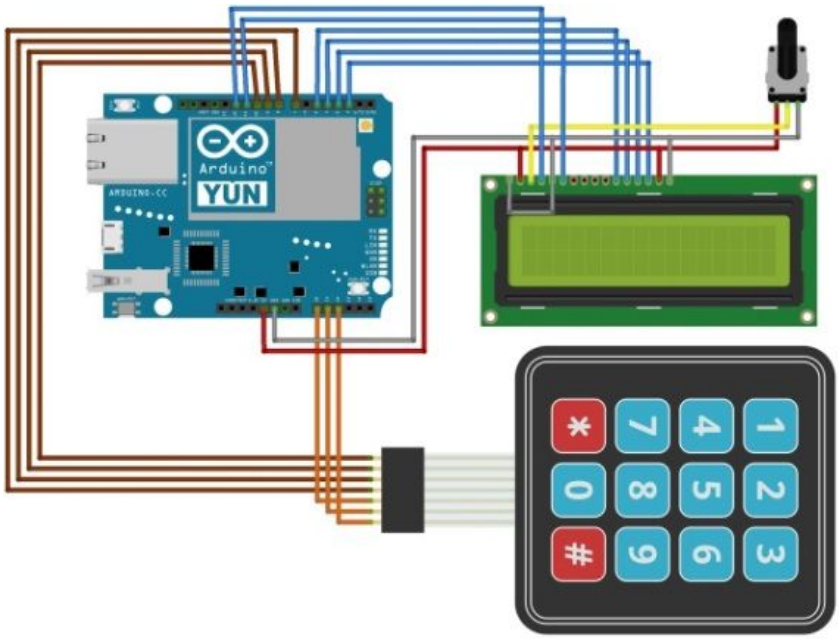
Dispositivo de captura de códigos de barras.

Dispositivo de despliegue de la información que está siendo capturada.

En el anexo 2 “Ficha técnica Arduino Yun”, se encuentran los modelos y precios de las partes y componentes utilizados para el armado del prototipo.

Para describir las conexiones necesarias entre dichos componentes, se utilizará un diagrama esquemático, el cual se enfoca en presentar las conexiones entre componentes, no su disposición o posición física real (16).

Figura 19 Diseño esquemático de los componentes del prototipo de hardware



Nota. Elaboración propia

En la figura 19 se observan las conexiones físicas realizadas entre la placa microcontroladora Arduino y los dos componentes principales: el display LCD y el teclado numérico.

Para la conexión del display LCD es necesario alimentarla con 5v, provistos por la placa microcontroladora (mostrados en color rojo) y utilizar un potenciómetro (conexión en color amarillo) de 10 KOhm para controlar a voluntad el brillo de la pantalla (22). De acuerdo a la hoja técnica del display utilizado, para transferir datos puede hacerse con un bus de 8 o 4 bits; esta última opción envía los datos en dos fases (23), y es la que se utilizará para economizar los pines digitales disponibles. Los pines de transferencia de datos de la pantalla se muestran en color azul.

Se utiliza un teclado numérico de membrana, el cual funciona como un arreglo de 4 columnas y cuatro filas, con un interruptor en cada botón que permite detectar cambios en el estado de las intersecciones para determinar qué tecla está siendo presionada. Este teclado es fácil de utilizar y de bajo precio (24); utiliza 8 pines del Arduino, cuyas conexiones se muestran en color café en la figura 17.

Tabla 9 Descripción de conexiones de Arduino con display LCD

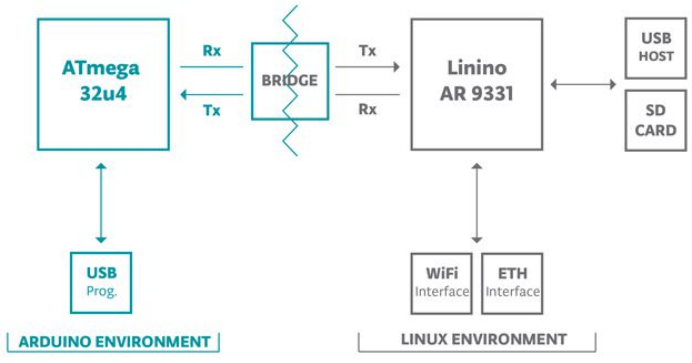
Pin Pantalla	Descripción	Pin Arduino
1	Ground.	GND
2	Voltaje para el control del LCD.	5V
3	Ajuste de voltaje (entrada a potenciómetro).	
4	Señal de selección de registros.	12
5	Definición de modo (Alto: lectura, bajo: escritura).	GND
6	Señal para escribir o leer datos.	11
7	Bus de datos (no utilizado)	
8	Bus de datos (no utilizado)	
9	Bus de datos (no utilizado)	
10	Bus de datos (no utilizado)	
11	Bus de datos (4 bits)	2
12	Bus de datos (4 bits)	3
13	Bus de datos (4 bits)	4
14	Bus de datos (4 bits)	5
15	Voltaje para la luz de fondo.	5V
16	Ground para luz de fondo.	GND

6.1.4 Captura de códigos de barra

Una de las ventajas que tiene el Arduino Yun, es que posee un puerto USB host de fábrica. Los dispositivos conectados a este son reconocidos por el sistema operativo Linino, y los datos capturados pueden ser enviados hacia el sketch.

Como se muestra en la figura 20, el Yun posee dos procesadores, y la comunicación entre ellos se realiza mediante la librería “Bridge”, dando a los sketches de Arduino, la habilidad de correr scripts del Shell de Linux, comunicarse con las interfaces de red y recibir información desde el procesador AR9331 (Linux). El puerto USB, las interfaces de red y la tarjeta SD no están conectados al 32U4 (Microcontrolador), pero el AR9331 y la librería Bridge le permite comunicarse con los periféricos (25).

Figura 20 Arquitectura con 2 procesadores del Arduino Yun



Nota. Tomado de (25)

Dado que el sistema operativo Linino es una variante de OpenWrt, que está orientado a routers, el soporte de dispositivos de Interfaz Humana (HID), como los USB, no está habilitado

de manera predeterminada. Para poder utilizar la funcionalidad USB, además de habilitar el soporte HID, se debe hacer un poco más de trabajo, lo cual incluye:

- Instalar los paquetes del kernel para el soporte de dispositivos HID.
- Capturar los eventos ejecutados por el dispositivo, los cuales se reciben de manera hexadecimal y convertirlos en caracteres entendibles para el humano (números en este caso).
- Crear un script que espere por estos eventos y los envíe vía Bridge al microcontrolador 32U4; el anexo “Script puente para eventos USB”, contiene el código que ejecuta el envío de la información ingresada a partir de un dispositivo USB, elaborado a partir de un script que identifica eventos de un ratón (26).

6.1.5 Conectividad Inalámbrica

Una de las ventajas que proporciona el Arduino Yun, es que la comunicación en red puede realizarse a través del puerto Ethernet o por medio del Wifi que tiene incorporado. Al utilizar el Wifi, la solución puede ganar movilidad, al igual que ahorrar costos generados por la instalación de cableado de red, que en las áreas de producción puede ser inadecuado si se sigue una política de manufactura esbelta, la cual trata de mantener únicamente objetos o materiales que son esenciales para el funcionamiento del área productiva y deshacerse de los demás (27).

Sin embargo, el prototipo utilizará información que puede ser sensible dentro de la empresa, y el hecho de conectarlo a la red interna a través de una red inalámbrica impone riesgos que deben tomarse en cuenta para evitar intrusiones, robo de información, entre otros.

Hay dos funciones de seguridad que hay que considerar en la protección de la red: La encriptación, que se refiere al cifrado criptográfico que sufren los datos al ser transmitidos por la red; y la autenticación, que es el acto de verificar que el dispositivo o persona que se conecta a la red es alguien autorizado (28).

Es recomendable que se utilice seguridad inalámbrica WPA2-Enterprise, ya que ofrece técnicas de encriptación aleatorias, la clave para desencriptar los datos es cambiada regularmente, por lo que si alguien descifra dicha clave, no será útil para futuras transferencias. Al utilizar este tipo de seguridad, es necesario establecer un servidor RADIUS, con el cual se puede tener una contraseña por cada dispositivo o usuario que se conectará a la red, en lugar de una sola clave compartida para todos. Además, es el mismo servidor RADIUS el encargado de la actualización de la clave de seguridad para encriptación (28).

Algunos modelos de puntos de acceso implementan un servidor RADIUS, pero en caso de no contar con un punto de acceso con estas características, es necesario configurar un servidor independiente, lo cual está fuera del alcance de este documento.

Es importante notar que para implementar la seguridad basada en WPA2-Enterprise, tanto el punto de acceso, como los dispositivos cliente deben ser capaces de soportar esta configuración. En el Arduino Yun, el sistema operativo Linino es quien se encarga de administrar las conexiones inalámbricas, para ello necesita tener instalado el paquete llamado WPAD. La configuración de fábrica no permite conectarse a una red empresarial ya que el paquete instalado es una versión reducida para conectividad básica, llamado WPAD-MINI; es necesario removerlo e instalar el paquete completo.

Puede encontrar una guía detallada de cómo realizar este proceso en el Anexo 2 Guía de software Arduino Yun, donde se detalla la secuencia de comandos necesarios para la instalación y la configuración del paquete; además, la configuración que debe realizarse en el archivo /etc/network/wireless para asociarse a una red con seguridad WPA2-Enterprise.

6.1.6 Alimentación eléctrica

Como en todo dispositivo electrónico, una fuente de alimentación estable es necesaria para su buen funcionamiento. La tabla 11 muestra el detalle del consumo promedio de los componentes que posee el prototipo:

Tabla 10 Consumo de componentes del prototipo

Componente	**Especificación**	**Consumo promedio**
Arduino Yun	Conectado a la red por medio de Wifi.	300 mA
Pantalla LCD 16x2 Hitachi.	Conectado a 5v del Arduino, dato tomado de la hoja de especificaciones.	3 mA
Escáner de códigos de barra.	Conectado al puerto USB Host del Arduino, asumiendo laser de detección de movimiento encendido.	110 mA

Los datos de la tabla anterior muestran que el prototipo tendrá un consumo promedio de 400 miliamperios utilizando el display LCD, lector de código de barras y conectividad vía Wifi. Este consumo puede aumentar de acuerdo a la actividad que tenga el prototipo, o disminuir si algunos componentes, como el escáner se encuentran inactivos.

Figura 21 Fuente de alimentación de 5.25V y 1A

Para un cumplir con los requisitos eléctricos del prototipo, se recomienda el uso de una fuente de alimentación regulada AC/DC de 5V (voltios) y 1A (amperio), aunque es más seguro utilizar una fuente de 5.25V y 2A, como la de la figura 21, si se desea conectar más periféricos al Arduino Yun (29). Obviamente, se debe tener en cuenta que el Yun sólo posee un puerto USB Host, y si se quieren utilizar más periféricos debe utilizarse un Hub USB multipuerto, con alimentación externa de preferencia.

Figura 22 Cargador USB portátil

Para darle movilidad a la solución, se puede hacer uso de baterías. Existen varias opciones en el mercado; en las pruebas de la investigación, se utilizó cargador portátil compacto, como el de la figura 22, que provee una salida de 5V y 1A; y tiene una capacidad de 2A; es decir, puede proveer 2A por una hora, lo que representa un aproximado de 3.5 a 4 horas para el prototipo. Existen cargadores más grandes y potentes que pueden durar el doble o el triple de esa cantidad, pero son más costosos y de mayor tamaño.

6.2 Diseño del software embebido

En este apartado, se detallan los requerimientos de software que se ejecutarán desde el Arduino, que tendrá la función de permitir al operario seleccionar la acción a realizar, solicitar los datos necesarios y enviar la información hasta la API.

Las siguientes son consideraciones para la codificación del programa Arduino:

Un sketch, o programa Arduino se compone de tres secciones: Sección de variables globales; la sección “void”, que se ejecuta una sola vez al momento de encender el Arduino; y la sección “loop” que se ejecuta de forma continua e ilimitada hasta que el Arduino se reinicie o se apague (30).

Dado que la sección loop, se puede ejecutar cientos o miles de veces por segundo, las variables declaradas localmente en esta sección perderán sus valores en cada ciclo; por lo tanto los datos ingresados por los usuarios deberán guardarse globalmente.

Los datos ingresados por el operario, las solicitudes y respuestas de la API, los mensajes desplegados para el usuario, entre otros; son datos que deben ser manejados por variables tipo

cadena (String), que deben ser utilizados cuidadosamente dentro del sketch. Esto se debe a que el alojamiento de cadenas se realiza en la SRAM (Memoria Estática de Aleatorio), la cual es limitada (2 KB en el Arduino Yun).

Por lo anterior es recomendable optimizar en el uso de cadenas literales y variables. Los literales deben ser almacenados en la memoria de programa (PROGMEM), en lugar de ser cargadas a la SRAM, dado su valor nunca cambiará durante la ejecución. Las variables que tienden a crecer durante la ejecución del programa pueden ser reservadas en memoria para evitar la fragmentación (31). Una prueba realizada con el sketch cargado al prototipo, demostró que se puede reducir hasta un 30% el uso de la SRAM aplicando estas técnicas.

6.2.1 Estructura del programa

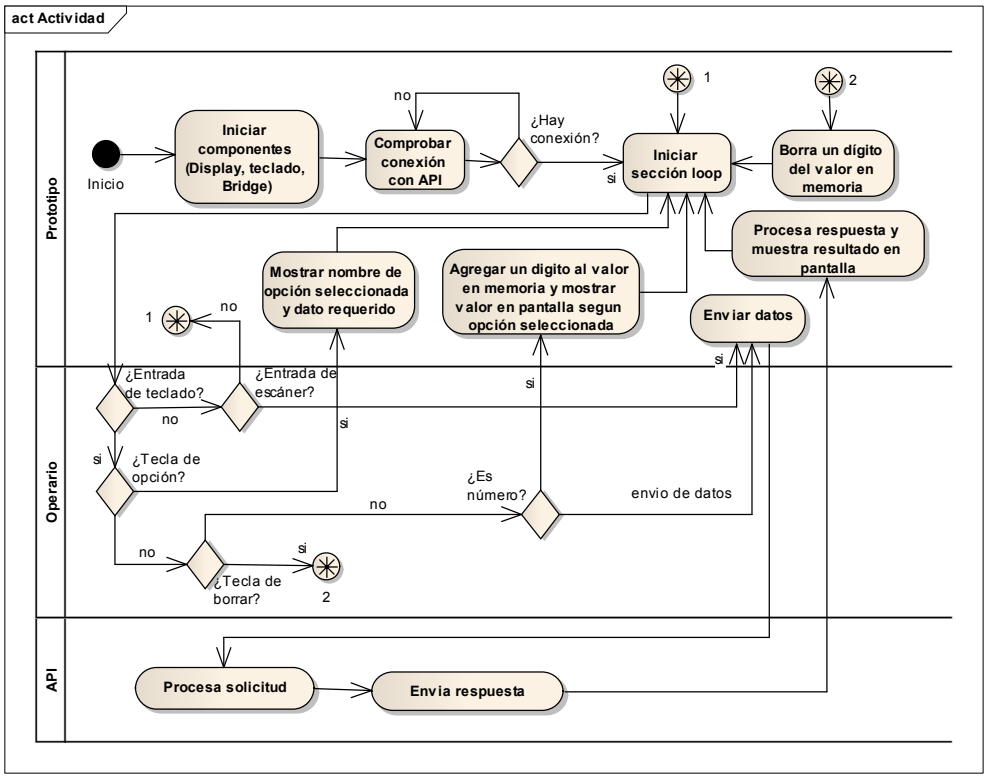
El programa cargado al prototipo debe cumplir con las características definidas en la especificación de requerimientos de software, puntualmente las historias de usuario: Iniciar proceso, Pausar proceso, Terminar Proceso e Ingreso a bodega.

En la figura 23, se muestra el flujo de procesos del sketch que se cargará al Arduino. En el diagrama se observa que se debe configurar inicialmente los componentes de captura y despliegue de información; y posteriormente verificar la conectividad con la API, el programa no continuará el flujo mientras no detecte conexión con el servicio donde al API está desplegada.

Dado que el operario puede interactuar con el prototipo mediante el teclado de membrana o mediante el escáner de códigos de barra, el programa verificará en cada ciclo loop si una tecla ha sido presionada; de no ser así, verificará si hay datos ingresados desde el dispositivo USB. Esto es así porque a través del teclado de membrana puede presionarse una sola tecla a la vez, mientras que con el escáner se reciben datos completos, por ejemplo un código de orden de fabricación o de empleado.

Por lo tanto, si el operario presionó una tecla, el programa debe determinar si dicha tecla es una opción, un dígito, la acción de borrar o la acción de enviar. Las opciones están representadas por las teclas A, B, C y D (y tendrán asociadas una de las historias de usuario anteriores); los dígitos, que son los números del cero al nueve, representan una parte de un código (de empleado, OF, artículo, etc.); el botón de borrar remueve el último carácter digitado; tecla de enviar, que indica que los datos ya están listos para ser validados y procesados. La figura 24 muestra las funciones de cada uno de los botones del teclado de membrana.

Figura 23 Diagrama de procesos que debe ejecutar el sketch de Arduino



Al presionar una tecla de opción, el programa solicitará el dato requerido para continuar con la acción, según lo definido en la historia de usuario. Por ejemplo, al presionar la opción iniciar proceso, el programa solicitará el código de empleado que realizará el trabajo; el usuario ingresará el código por medio del teclado de membrana o escaneará un código de barras para que la API verifique si es un empleado válido; de ser así solicitará el número de la OF, la cual también será validada, y posteriormente se ejecutará el proceso correspondiente.

Figura 24 Funciones del teclado de membrana

La tecla debe ser presionada por el operario para indicar que la información ingresada está completa y debe ser validada o procesada. Es decir, si el programa le solicita el código de

empleado, el usuario debe presionar la tecla enviar cuando haya digitado su código. En el caso de hacerlo con código de barras, los escáner normalmente están configurados para enviar como parte de la cadena el carácter “entrar”, con lo que se ejecutaría automáticamente el envío de datos hacia la API.

El anexo 4“Firmware del prototipo” contiene el código que será cargado al microcontrolador del Arduino, es una versión funcional, resultado de la investigación actual.

6.2.2 Comunicación con la API

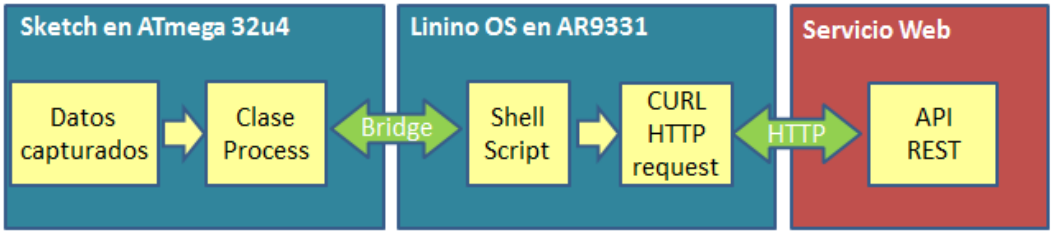
El diseño de la solución para la comunicación del prototipo con la API, tiene como objetivo desacoplar ambas partes; de esta forma el prototipo no necesita que la API esté escrita en un lenguaje específico, o desplegada en una arquitectura en particular. Asimismo, la API no debe requerir que los datos que recibe sean enviados desde un dispositivo específico, es decir, el prototipo no necesita ser construido necesariamente con Arduino.

La solución hará uso de HTTP (Protocolo de Transferencia de Hipertexto), el cual es un protocolo para sistemas de información diseñado para esconder los detalles de cómo un servicio es implementado, presentando una interfaz uniforme. HTTP fue creado para la web, y funciona intercambiando mensajes enviados entre un cliente y un servidor. Un cliente es un programa que establece conexión con un servidor con el propósito de enviar una solicitud (request); un servidor, es un programa que acepta conexiones y brinda servicio a una solicitud, enviando respuestas HTTP (response) (32).

Si se utiliza una placa Arduino UNO o Mega, puede realizar conexiones y solicitudes HTTP por medio de las librerías EthernetClient, si está utilizando la interfaz Ethernet; y WifiClient, si utiliza conexión inalámbrica. Pero, ya que se está utilizando un Arduino Yun, la comunicación se realizará desde el sistema operativo (Linino), a través de la utilidad CURL.

CURL es una herramienta para línea de comandos que permite transferir datos o archivos desde y hacia un servidor utilizando sintaxis basada en URL, y que soporta protocolos HTTP, HTTPS, FTP, entre otros (33). Viene instalado de fábrica en Linino.

Figura 25 Esquema de comunicación entre el prototipo y la API



Debido a que el Arduino Yun posee dos procesadores, para trasladar los datos capturados desde el sketch hacia el lado del sistema operativo, se requiere la librería “Process”, la cual permite ejecutar comandos Linux alojados del lado del SO desde el procesador 32u4 (34). Esto permitirá ejecutar scripts del Shell de Linux que realizarán la comunicación con la API, a través del comando CURL, y procesar la respuesta enviada por el servidor a través de la API.

La figura 25 muestra un esquema de comunicación simplificado entre el prototipo y la API, donde el sketch se comunica vía la clase Process con un script del Shell de Linux que ejecuta el comando CURL para realizar una solicitud HTTP al servidor que despliega la API, mostrando una interfaz REST, es decir, consumida a través de URLs. El diagrama de despliegue muestra la arquitectura completa sugerida para el despliegue de la solución.

6.2.3 Descripción de librerías utilizadas

El programa cargado al prototipo, hará uso de varias librerías ya existentes, para controlar componentes, o realizar procesos de comunicación. La tabla 11, muestra las librerías utilizadas y su función:

Tabla 11 Librerías utilizadas en firmware de prototipo

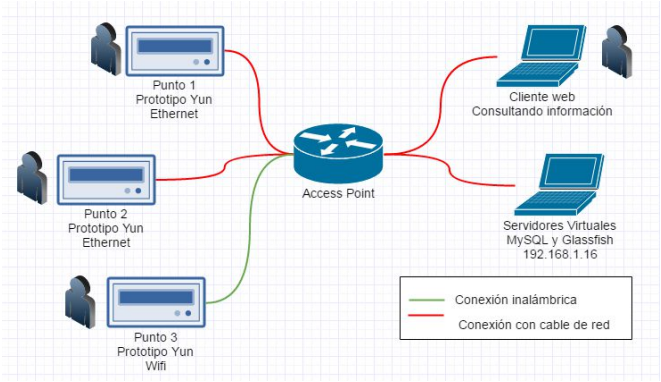
Librería	Descripción
Keypad.h	Es una librería para el uso de teclados matriciales con el Arduino. La versión 3.0 ya soporta la presión de múltiples teclas.
LiquidCrystal.h	Permite controlar display de cristal líquido (LCD) basado en el chip Hitachi HD44780 (o compatible). La librería funciona en modo de 4 o 8 bits.
Bridge.h	Es específica del Arduino Yun. Permite y simplifica la comunicación entre el ATmega 32u4 y el AR93331. Los comandos enviados desde el microcontrolador son interpretados por Python en el AR9331. Permite comunicación en ambas direcciones y actúa como interfaz con la línea de comandos de Linux.
Process.h	Es una clase que permite correr procesos y comandos de Linux desde el ATmega 32u4.

7 Resultados

7.1 Prueba inicial

El modelo propuesto fue probado en un ambiente controlado, desplegando la solución como se observa en la figura 26. Se instalaron servidores virtuales, utilizando Oracle VirtualBox, donde se desplegó el servidor Glassfish 4.0 y MySqL 5.7.

Figura 26 Escenario de prueba de prototipo y API



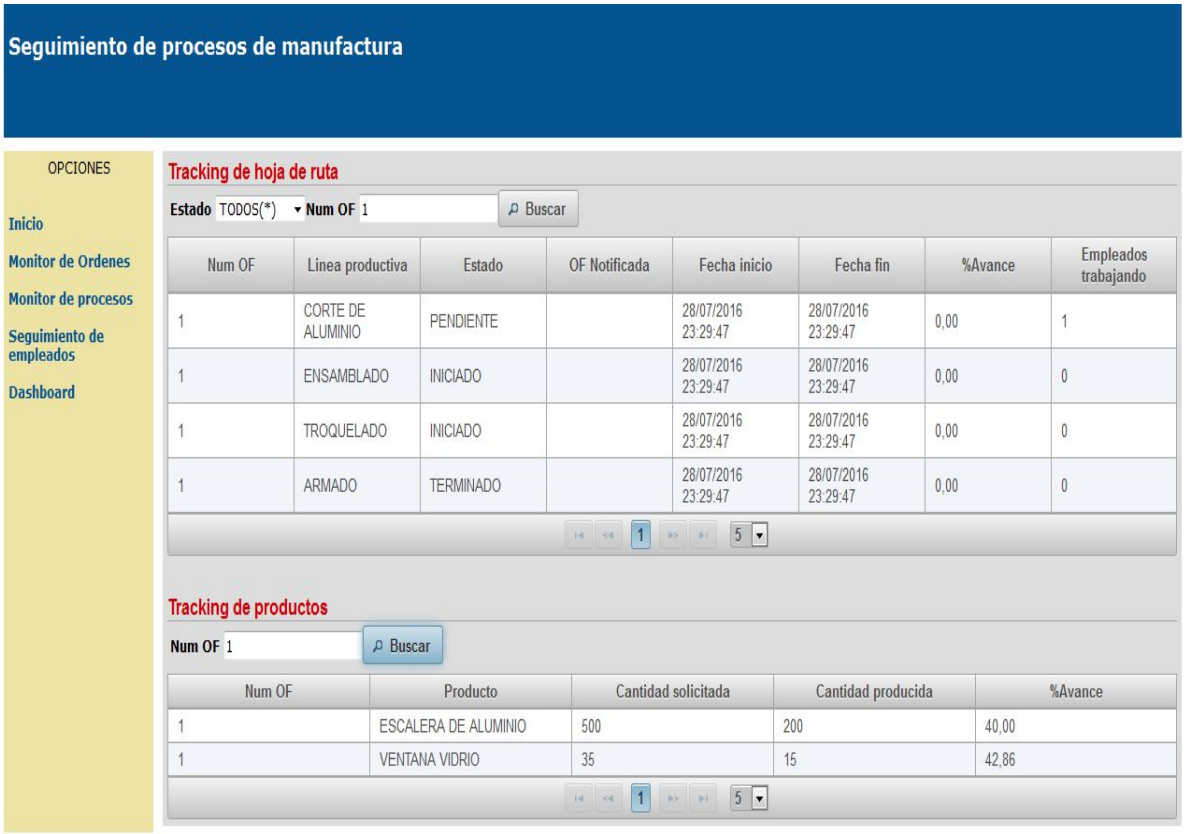
Nota. Elaboración propia

Además, se crearon tres prototipos basados en Arduino Yun, mostrados en la figura 27, de los cuales dos se conectaron de forma cableada por el puerto Ethernet a la red local y uno se conectó de forma inalámbrica.

Figura 27 Prototipos utilizados en la prueba inicial

Se elaboró un proyecto web en Java, utilizando JPA para la persistencia e implementación de la lógica de la aplicación; y, Prime Faces para las páginas web de prueba donde se realiza el seguimiento en tiempo real del trabajo realizado por los operarios. La Figura 28 muestra la hoja de ruta de una orden de fabricación, y los avances registrados desde los puntos de control.

Figura 28 Pantalla de seguimiento realizada para la prueba inicial.

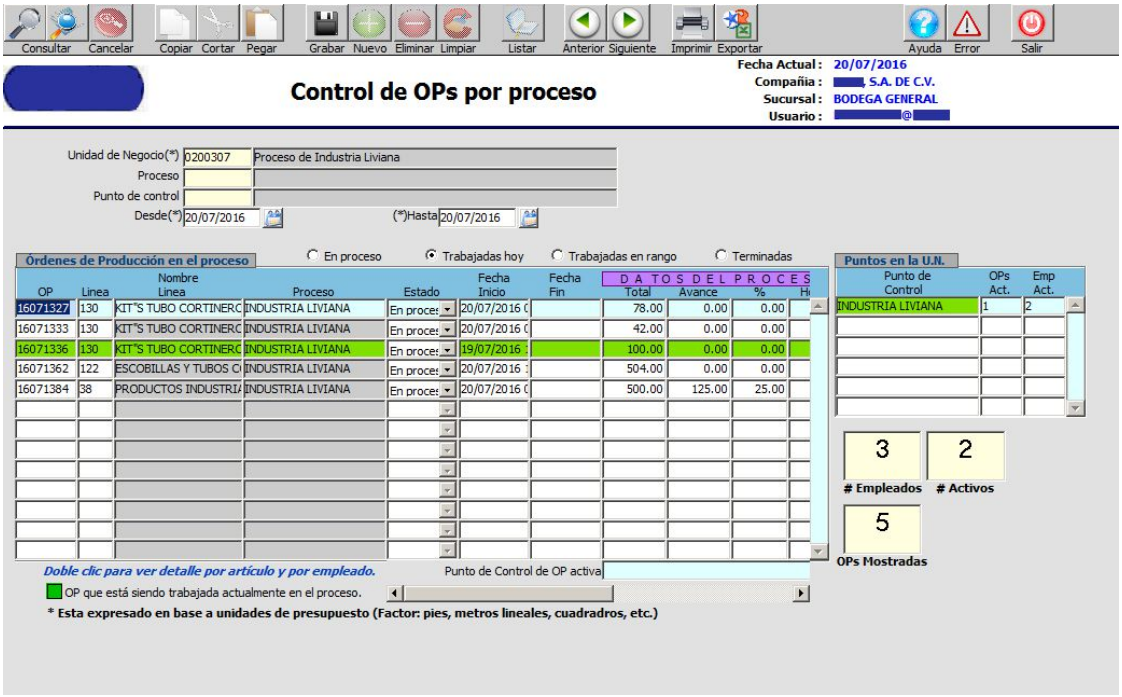


7.2 Prueba piloto en fábrica

Este modelo de control de piso además ha sido instalado exitosamente en una empresa dedicada a la manufactura de productos de aluminio, que realiza órdenes de fabricación de alto y bajo volumen. Su catálogo de artículos es variado y abarca productos de fabricación estándar, productos a la medida y bajo demanda. La empresa cuenta con una planilla que ronda los 250 empleados, de los cuales un 80% pertenecen a la fábrica, y el resto a las áreas administrativas, mantenimiento, entre otros.

Originalmente, para el cálculo de los costos de mano de obra, cada operario en la planta reportaba al final del día en una hoja de control, las órdenes de fabricación trabajadas con un estimado de tiempo invertido en cada una. Además, los ingresos de los productos terminados a bodega se hacían en un formato impreso, que posteriormente se digitaba en el sistema interno.

Figura 29 Pantalla de consulta de órdenes de fabricación en proceso en un departamento productivo



Esta forma de controlar los procesos tiene los siguientes inconvenientes:

- El tiempo de mano de obra asignada a órdenes de fabricación es inexacto.
- No se conoce cuál es el trabajo en proceso, a menos que se consulte verbalmente con el supervisor de la producción.
- Los ingresos a bodega generan pérdida de tiempo y exceso de desplazamiento.

La solución propuesta se implementó utilizando puntos de control en una zona de la fábrica, todos conectados de forma inalámbrica a un punto de acceso existente, con capacidad de validación WPA2-Enterprise.

La implementación de la solución permite reducir el tiempo que los supervisores invertían cargando los datos de mano de obra para cada orden de fabricación. Este tiempo se redujo de un promedio de 30 minutos con la metodología anterior, a 5 minutos con la solución propuesta. En los ingresos a bodega dependiendo de la cantidad de artículos, se ha reducido el tiempo entre un 60 y 75%, debido a la disminución del papeleo y desplazamientos.

Figura 30 Prototipo instalado en la fábrica

La figura 30 muestra el punto de control instalado en uno de los departamentos; el prototipo cuenta con un display, teclado, una carcasa protectora y un lector de código de barras en la parte inferior conectado al puerto USB host.

La figura 29 muestra la pantalla de consulta de las órdenes de fabricación trabajadas durante el día en un departamento específico.

7.3 Inconvenientes

La recepción de la señal inalámbrica en el prototipo puede verse afectada si se ubican carretas con material entre el dispositivo y la antena o punto de acceso. En la empresa donde se implementó se utiliza aluminio, lo cual puede interferir con la señal y hacer necesario un reinicio del prototipo si este queda bloqueado, esta situación ha sucedido con poca frecuencia.

Otro inconveniente es el tiempo que tarda el Arduino Yun en cargar completamente el sistema operativo. El prototipo tarda un aproximado de un minuto y medio para poder ser utilizado, esta situación ocurre por la mañana al iniciar labores.

8 Conclusiones

En este documento se ha presentado una propuesta de diseño de una API REST para el control de procesos de manufactura, los cuales incluyen seguimiento en tiempo real del trabajo en proceso y avances de órdenes de fabricación, control de tiempos de mano de obra, que son un insumo para el cálculo del costo real de los artículos fabricados. Esta información, capturada

desde un dispositivo construido a partir de microcontroladores Arduino, los cuales permiten desarrollar una tarea específica de forma eficiente y a bajo costo.

Este trabajo tiene las siguientes contribuciones: Primero, la elaboración de un conjunto de requerimientos, basado en la investigación bibliográfica y recopilación de necesidades en una empresa local. Segundo, se propone un diseño Orientado a Objetos, documentado con UML que sirve como insumo para el desarrollo en la tecnología de nuestra elección. Tercero, un diseño de un prototipo de hardware, basado en Arduino Yun, que ha sido validado en un entorno de producción real y que es capaz de consumir Web Services, por lo que utilizado con la API propuesta, podría ser integrado con sistemas existentes.

Los siguientes pasos incluyen, mejorar la velocidad de inicio del prototipo, no se descarta probar otro modelo de microcontrolador; el diseño de una carcasa más estética para que sea más atractivo para pequeñas y medianas industrias en el país; y, el uso de escáner inalámbrico para la lectura de código de barra.

9 Recomendaciones

La notificación de órdenes de fabricación asignadas al punto de control se realiza con un mensaje mostrado en la pantalla del dispositivo. Un diseño posterior puede integrar un buzzer o zumbador que emita un sonido al detectar un nuevo trabajo enviado al piso de producción.

El porcentaje de avance mostrado en la pantalla de seguimiento por orden de fabricación, toma como referencia la cantidad de procesos finalizados respecto al total de procesos de la OF. El avance por proceso mostrado en la figura 13, es 100% si el proceso ya está completo y 0% si dicho proceso está incompleto; y el avance por producto, se calcula con la cantidad de producto solicitado y el total de producto terminado. En algunas industrias puede ser necesario un control más detallado de los avances realizados en cada proceso, tomando en cuenta la cantidad de piezas trabajadas, sub ensambles listos, entre otros. Para futuras investigaciones, se sugiere implementar este tipo de seguimiento, considerando además que la captura de información no debe ser causa de disminución en la productividad de los operarios.

La configuración del número de punto de control y la dirección IP del servidor donde se despliega la API, se realiza vía SSH en archivos planos del lado del Sistema Operativo del Arduino. Se recomienda el desarrollo de un método de configuración simplificado e intuitivo, a fin de no requerir de un experto para realizar la configuración inicial.

10 Bibliografía

1. **Zigbee Alliance.** Zigbee White Papers. *sitio web de Zigbee Alliance.* [En línea] Julio de 2009. [Citado el: 31 de Julio de 2016.] http://web.archive.org/web/20100411233226/http://www.zigbee.org:80/LearnMore/WhitePapers.aspx.

2. **Kun-Yung, Lu.** *Implementing a real-time shop-floor control system using zigbee-based networking systems.* National United University, Miaoli, Taiwan : 2013.

3. **Moreira, Luciana, y otros.** SOCRADES: A Web Service based Shop Floor Integration Infrastructure. Zurich : s.n., 2008.

4. **SAP.** SAP xMII System Overview. *sitio de Ayuda de SAP Corporation.* [En línea] 2007. [Citado el: 31 de Julio de 2016.] https://help.sap.com/saphelp_xmii115/helpdata/en/Introduction/IllumSystemOverview.htm.

5. **Serope Kalpakjian, Steven R. Schmid.** *Manufactura, Ingeniería y Tecnología.* Quinta edición. México : Pearson Education, 2008.

6. **Groover, Mikell P.** *Fundamentos de manufactura moderna.* México : McGraw-Hill, 2007.

7. **Chapman, Stephen N.** *Planificación y control de la producción.* México : Pearson Educación, 2006.

8. **SAP.** Órdenes de Fabricación. *SAP.* [En línea] [Citado el: 29 de Noviembre de 2015.] http://help.sap.com/saphelp_470/helpdata/es/97/1648e309e011d3b6b30000e8359890/content.htm.

9. **Reyes, Carlos A.** *Microcontroladores PIC Programación en Basic.* Quito : Rispergraf, 2008. 9978-45-004-1.

10. **Angulo, José María.** *Microcontroladores PIC Diseño práctico de aplicaciones.* Madrid : McGraw-Hill, 2003. 84-481-3788-4.

11. **Craft, Brock.** *Arduino Projects.* Chichester, Inglaterra : John Wiley & Sons, Ltd, 2013.

12. **Pfister, Cuno.** *Getting Started with the Internet of Things.* Sebastopol, California : O'Reilly, 2011. 978-1-4493-9357-1.

13. **Joyanes, Luis.** *Computación en la nube.* Mexico D.F. : Alfaomega, 2012. 978-607-707-468-7.

14. **McEwen, Adrian y Cassimally, Hakin.** *Designing the Internet of Things.* United Kingdom : John Wiley & Sons Ltd, 2014. 978-1-118-43065-1.

15. **Arnowitz, Jonathan.** *Efective Prototyping for Software Designers.* San Francisco, California : Morgan Kaufmann Publishers, 2007.

16. **Margolis, Michael.** *Arduino cookbook.* California : O'reilly, 2011.

17. **Raspberry Pi Foundation.** About us. *sitio web de Raspberry Pi Foundation.* [En línea] [Citado el: 2016 de Enero de 27.] https://www.raspberrypi.org/about/.

18. **Bloch, Joshua.** *Effective Java.* Santa Clara, California : Addison-Wesley, 2008. 978-0-321-35668-0.

19. **Microsoft.** XML Web Services Overview. *Microsoft Developer Network.* [En línea] [Citado el: 25 de Enero de 2016.] https://msdn.microsoft.com/en-us/library/w9fdtx28(v=aspnet.11).aspx.

20. **Massé, Mark.** *REST API Design Rulebook.* Sebastopol, California : O'Reilly, 2012.

21. **Kniberg, Henrik.** *SCRUM Y XP DESDE LAS TRINCHERAS.* Estados Unidos de América : s.n., 2007.

22. **Arduino.** Arduino - Hello World. *Arduino.* [En línea] 17 de 08 de 2015. [Citado el: 2016 de Mayo de 01.] https://www.arduino.cc/en/Tutorial/HelloWorld.

23. **Adafruit.** Standar LCD 16x2. *Adafruit.* [En línea] [Citado el: 2016 de Mayo de 01.] https://www.adafruit.com/products/181.

24. **Parallax Incorporated.** 4x4 Membrane Keypad. *sitio web de Parallax.* [En línea] 16 de Diciembre de 2011. [Citado el: 1 de Mayo de 2016.] https://www.parallax.com/sites/default/files/downloads/27899-4x4-Matrix-Membrane-Keypad-v1.2.pdf.

25. **Arduino.** Arduino Yun. *Arduino.* [En línea] [Citado el: 1 de Mayo de 2016.] https://www.arduino.cc/en/Guide/ArduinoYun.

26. **Dicola, Tony.** GitHub: Evil Mouse Prank. *GitHub.* [En línea] [Citado el: 2016 de Junio de 26.] https://github.com/tdicola/EvilMousePrank.

27. **Sayer, Natalie y Bruce, William.** *Lean for dummies.* Indianapolis : Wiley Publishing Inc, 2007.

28. **Briere, Danny y Hurley, Pat.** *Wireless Home Networking .* Indianapolis : Wiley Publishin Inc., 2011.

29. **Adafruit Industries.** 5V 2A SWITCHING POWER SUPPLY W/ USB-A CONNECTOR. *sitio web de Adafruit Industries.* [En línea] [Citado el: 21 de Junio de 2016.] https://www.adafruit.com/products/1994.

30. **Torrente, Oscar.** *Arduino: Curso práctico de rormación.* Madrid : Alfaomega, 2013.

31. **Adafruit Industries.** Optimizing SRAM. *sitio web de Adafruit Industries.* [En línea] 17 de Abril de 2016. [Citado el: 08 de Junio de 2016.] https://learn.adafruit.com/memories-of-an-arduino/optimizing-sram.

32. **W3C.** Hypertext Transfer Protocol. *W3C Architecture Domain.* [En línea] Junio de 2014. [Citado el: 2016 de Junio de 14.] https://tools.ietf.org/html/rfc7230.

33. **curl.** curl and libcurl. *curl.* [En línea] [Citado el: 16 de Junio de 2016.] https://curl.haxx.se/.

34. **Arduino.** Arduino - Process. *Arduino.* [En línea] 25 de Mayo de 2016. [Citado el: 14 de Junio de 2016.] https://www.arduino.cc/en/Tutorial/Process.

35. **Horngren, Charles, Datar, Srikant y Rajan, Madhav.** *Contabilidad de Costos un enfoque gerencial.* Decimocuarta. Naucalpan de Juárez : Pearson, 2012. 9780132109178.

36. **Barr, Michael y Anthony, Massa.** *Programming Embedded Systems with C and GNU Development Tools.* Sebastopol : O'reilly, 2007.

37. **Banco Central de Reserva de El Salvador.** Producto Interno Bruto (PIB) por rama de actividad económica. *sitio web MINEC.* [En línea] [Citado el: 9 de Julio de 2016.] http://www.bcr.gob.sv/bcrsite/?cdr=30.

38. **World Economic Forum.** Global Information Technology Report. *Reports - World Economic Forum.* [En línea] 2015. [Citado el: 05 de Julio de 2016.] http://reports.weforum.org/global-information-technology-report-2015/economies/#economy=SLV.

11 ANEXOS

11.1 Evaluación de placas microcontroladoras

				Comparación				Evaluación				Ponderación			
ID	Criterio	Descripción	Peso	Arduino Yun	Raspberry Pi 2	Particle Photon	Beagle bone black	Arduino Yun	Raspberry Pi 2	Particle Photon	Beagle bone black	Arduino Yun	Raspberry Pi 2	Particle Photon	Beagle bone black
1	Procesador	Velocidad del microprocesador, MIPS (millones de instrucciones por segundo).	8	400 Mhz (para Linux)	1.2 GHz	120 Mhz	1GH	4	5	5	5	32	40	40	40
2	RAM	Cantidad de memoria RAM disponible para procesar los datos.	8	64 MB (para Linux), 2	1GB	128 KB	512 GB	4	5	4	5	32	40	32	40
3	Facilidad de uso	Facilidad de aprendizaje del lenguaje de programación y ensamblado de los componentes.	8	Arduino, Bash, Python Variedad de Shields, sensores y actuadores	Python (default) y posibilidad de instalar diversos compiladores. No hay soporte de tiempo real, menor potencia en computación física	Lenguaje Arduino y posibilidad de programar con NodeJS, consumo de web services y carga de código de manera inalámbrica, desde internet	Instalación de diversos compiladores. Programación de fábrica con NodeJs. Soporta tiempo real, pero no es sencillo.	4	4	5	5	32	32	40	40
4	Consumo de energía	Permite en conjunto con los demás componentes no exceder los 20 Voltios y 3.5 Amperios.	8	250 mA	375 mA	80 mA	315 mA	4	3	5	3	32	24	40	24
5	USB	Capacidad para conectar un dispositivo USB al microcontrolador.	8	1 puerto	4 puertos	No tiene capacidad de conectar un dispositivo USB esclavo.	1 puerto	4	5	2	4	32	40	16	32
6	Facilidad de conexión y ensamble	Es fácil de ensamblar y conectar con las demás partes del prototipo.	8	Conexión por medio de pines GPIO (14 digitales y 6 analógicos). Un puerto USB para conexión de lector de barras.	Conexión por medio de puertos USB (Teclado y scanner), Pantalla necesita un adaptador de RCA a video compuesto (hecho a mano).	Conexión por medio de pines GPIO (8 digitales y 6 analógicos). Necesita el shield shield y USB host shield para conectar escaner USB.	Conexión de pantalla con HDMI, teclado y escaner con USB, necesitará un HUB USB, ya que sólo tiene un puerto.	5	4	3	3	40	32	24	24

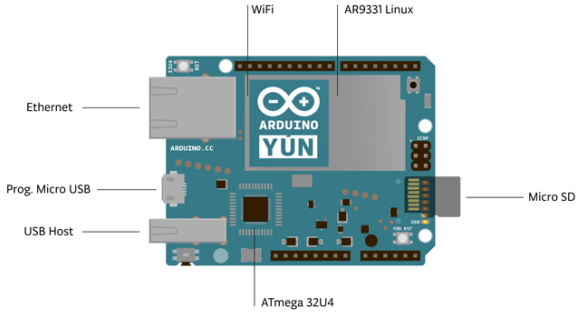
				Comparación				Evaluación				Ponderación			
ID	Criterio	Descripción	Peso	Arduino Yun	Raspberry Pi 2	Particle Photon	Beagle bone black	Arduino Yun	Raspberry Pi 2	Particle Photon	Beagle bone black	Arduino Yun	Raspberry Pi 2	Particle Photon	Beagle bone black
7	Costo	Mantiene el valor del prototipo relativamente bajo, incluyendo accesorios	10	105.53	133.19	111.71	154.27	5	4	5	3	50	40	50	30
8	Tamaño	Ayuda a mantener compacto el tamaño del prototipo.	8	Tamaños con case: alto: 12.7 cm ancho: 9.15 cm espesor: 3.4 cm 1 fuente de alimentación de 5v	Tamaño con pantalla TFT: alto: 11.3 cm ancho: 17.4 cm espesor: 5 cm 2 fuentes de alimentación (5 y 12v)	El chip mide 4x2 cm, es el más peque;o de todos. Aunque necesita shields adicionales para utilizar USB, lo cual aumenta el tamaño.	Tamaño con pantalla TFT: alto: 11.3 cm ancho: 17.4 cm espesor: 5 cm 2 fuentes de alimentación (5 y 12v)	4	3	5	3	32	24	40	24
9	Presencia en el mercado nacional	Existen distribuidores locales de la tarjeta microcontroladora.	10	SI	SI	NO	NO	5	5	2	2	50	50	20	20
10	Conectividad a red	Permite la conexión a red local o Internet de forma cableada e inalámbrica.	8	Ethernet + wifi	Ethernet	Wifi, con soporte de l	Ethernet	5	2	4	2	40	16	32	16
11	Madurez en el mercado	Cantidad de años en el mercado	8	Lanzada en 2004	Lanzada en 2012	Lanzado en 2014	Lanzada en 2008	5	4	3	4	40	32	24	32
12	Comunidad y soporte activo	Se encuentra soporte y foros activos en las páginas oficiales	8	Comunidad activa en la web. Open Hardware en El Salvador. 2675 entradas de blog en adafruit.com	Comunidad más grande con tutoriales y guías en linea. Open Hardware en El Salvador. 3685 entradas de blog en adafruit.com	Comunidad relativamente más reducida.	Comunidad relativamente más reducida.	4	5	3	3	32	40	24	24
		TOTAL	100									**444**	410	382	346

11.2 Ficha técnica Arduino Yun

NOMBRE DEL PRODUCTO	**Arduino Yun**	
EMPRESA	Arduino	

DESCRIPCION DE LA PLACA

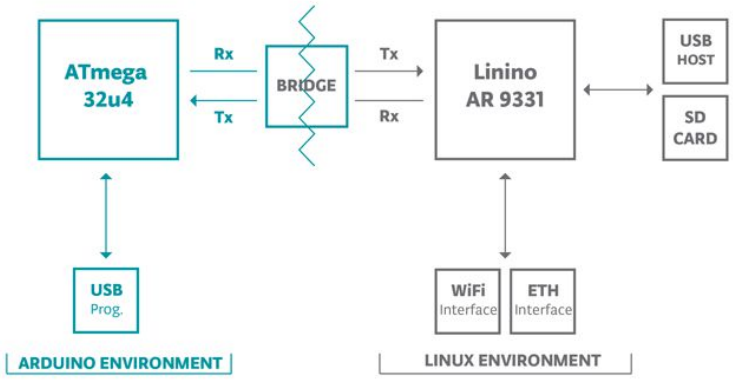
El Arduino Yun es una placa microcontroladora basada en ATmega32u4 y Atheros AR93331. El procesador Atheros soporta un distribución Linux basada en OpenWrt llamada OpenWrt-Yun. La tarjeta posee soporte de Ethernet y Wifi de fábrica, un puerto USB-A, ranura para tarjeta micro-SD, 20 pines digitales, de los cuáles 7 pueden ser utilizados como salidas de PWM y 12 como entradas digitales, tiene un oscilador de cristal de 16MHz, conexión micro USB y 3 botones de reinicio.



El Yun se distingue de otras tarjetas Arduino en que puede comunicarse con la distribución de Linux que trae en la tarjeta, ofreciendo una computadora con conectividad a red, combinada con el procesamiento de Arduino. Además de los comandos propios de Linux, como CURL, puede escribir sus propios scripts en Shell o Python para interacciones más robustas.

Arquitectura:

El Yun posee dos procesadores, y la comunicación entre ellos se realiza mediante la librería “Bridge”, dando a los sketches de Arduino la habilidad de correr scripts del Shell de Linux, comunicarse con las interfaces de red y recibir información desde el procesador AR9331. El puerto USB, las interfaces de red y la tarjeta SD no están conectados al 32U4, pero el AR9331 y la librería Bridge permite al Arduino comunicarse con los periféricos.



ESPECIFICACIONES TECNICAS

Microcontrolador

Característica	**Descripción**
Microcontrolador	ATmega32U4
Voltaje operativo	5V
Voltaje de entrada	5V
Pines digitales de entrada y salida	20
Canales de manejo de corriente PWM	7
Pines de entrada analógicos	12
Corriente directa por pin	40 mA
Corriente directa por pin 3.3V	50 mA
Memoria Flash	32 KB (4 usados para el bootloader)
SRAM	2.5 KB
EEPROM	1 KB
Velocidad del reloj	16 MHz
Consumo medio de energía	250 mA

Procesador Linux

Característica	Descripción
Procesador	Atheros AR9331
Arquitectura	MIPS@400 MHz
Voltaje de operación	3.3V
Ethernet	IEEE 802.3 10/100 Mbit/s
Wifi	IEEE 802.11b/g/n
Lector de tarjetas	Micro-SD
RAM	64 MB DDR2
SRAM	2.5 KB
EEPROM	1KB
Velocidad del reloj	16 MHz

Programación:

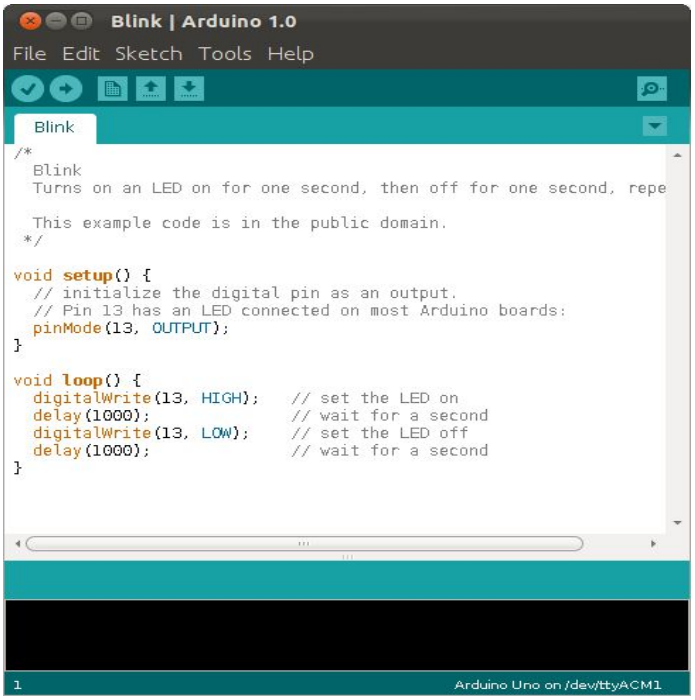
Para conectar el Arduino Yun a la computadora, se necesita un cable Micro-B USB. Este cable provee energía y datos a la tarjeta. Para programar en Arduino utiliza el IDE de desarrollo de Arduino en versión 1.5.4 o superior, este IDE puede ser utilizado desde Windows, Linux u OSX.

Del lado del procesador AR9331 corre la distribución de Linux para sistemas embebidos (OpenWrt-Yun), se incluye una instalación de Python 2.7, con la que se puede realizar programación avanzada, y comunicarse a través de la clase Bridge con el microcontrolador ATMega32U4.

La configuración de la distribución Linux se puede acceder desde una interfaz web ya que el Yun maneja Web Services REST. Cuando se conecta a la red, ya sea cableada o inalámbrica, puede programar el Arduino por medio de la red. El puerto Ethernet, de forma predeterminada se conecta vía DHCP.

El Arduino cuenta con la librería “Process” para invocar procesos de Linux, por lo que puede interactuar con scripts del Shell de Linux o Python.

El Arduino puede ser extendido a través de librerías, por lo tanto, es posible interactuar con una gran cantidad de componentes a través de las librerías que proveen los fabricantes.



Interacción con dispositivos:

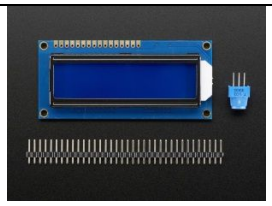
La plataforma Arduino cuenta con muchas librerías disponibles para utilizar con muchos dispositivos y facilitar su utilización. Entre ellas se encuentran:

Librería	Descripción
Bridge	Permite intercambiar información entre la distribución de Linux y el sketch de Arduino.
EEPROM	Permite la lectura y escritura a la memoria permanente de la tarjeta.
Ethernet	Permite la conexión (local e Internet) por medio del shield Ethernet.
Firmata	Permite la comunicación con apps de computadoras utilizando el protocolo serial.
GSM	Permite conexión a través de una red GSM o GRPS, envío y recepción de SMSs.
LiquidCrystal	Permite la comunicación con display LCD alfanuméricos.
SD	Permite la lectura y escritura a tarjetas SD.
Servo	Permite control una variedad de servomotores.

PRECIOS Y PROVEEDORES

Nombre	Datos	Precio
ARDUINO STORE USA	País: Estados Unidos. Método de pago: Electrónico (Paypal, tarjeta de crédito). Sitio web: https://store.arduino.cc/	$57.82
adafruit	País: Estados Unidos. Método de pago: Electrónico (Paypal, tarjeta de crédito). Sitio web: https://www.adafruit.com	$74.95
amazon.com	País: Estados Unidos. Método de pago: Electrónico (Paypal, tarjeta de crédito). Sitio web: http://www.amazon.com/	$66.58
teubi.co Tienda	País: El Salvador Sitio web: http://tienda.teubi.co/ Dirección: Avenida Olímpica, Colonia Escalón, Galería Olímpica #111	$95.00

ACCESORIOS Y COMPONENTES

Componente	Imágen	Proveedor	Precio
Display LCD 16x2		www.adafruit.com	9.95
Teclado de membrana		http://www.amazon.com/	3.88
Carcasa para Arduino		www.adafruit.com	15
Escaner de código de barras		http://www.amazon.com/	18.88
Total Accesorios			47.71
Total + Tarjeta			105.53

FUENTES

https://www.arduino.cc/en/Guide/ArduinoYun
https://www.arduino.cc/en/Main/ArduinoBoardYun
https://www.sparkfun.com/products/12053
https://learn.adafruit.com/embedded-linux-board-comparison/power-usage

11.3 Script puente para eventos USB

```
#!/usr/bin/python
import sys
import evdev
import os
import os.path
import select
import struct
import time
import traceback

# Script creado por Luis Arguijo a partir del Script de Tony Dicola
encontrado en:
# https://github.com/tdicola/EvilMousePrank
# Debe apuntar al teclado o escáner USB (que debe ser siempre event1) si sólo
# hay un dispostivo conectado al Yun.
# Este archivo debe ser ejecutado al iniciar el Arduino Yun.
# Debe ser invocado desde el archivo /etc/rc.local
# Agregar la siguiente linea:
# /ruta/KeyboardBridge.py &
# Con el ampersand al final para que se ejecute en segundo plano de manera
permanente.
DEVICE_PATH = '/dev/input/event1'
# Usar un archivo FIFO (First Input First Output), para recibir eventos de
otros proceso
FIFO_PATH = '/tmp/EvilMousePrank.fifo'


if __name__ == '__main__':
  while True:
    try:
      # Elimina el FIFO si ya existe.
      if os.path.exists(FIFO_PATH):
        os.remove(FIFO_PATH)
        print "Fifo"
      # Crea el FIFO e inicia comunicación de solo lectura.
      os.mkfifo(FIFO_PATH)
      print "os.mkfifo"
      # Al abrir el archivo FIFO, asegurar se especificar la bandera
O_NONBLOCK,
      # para evitar que la apertura del archivo se de hasta que algún
      # dispositivo se conecte.
      fifo = os.open(FIFO_PATH, os.O_RDONLY | os.O_NONBLOCK)
      # Cargar a las variables del sistema la ruta donde se encuentra el
bridge
      # hecho en python para la comunicación con el ATmega.
      print "brigde"
      sys.path.insert(0, '/usr/lib/python2.7/bridge/')
      # Importar la librería BridgeClient
      from bridgeclient import BridgeClient as bridgeclient
      bc = bridgeclient()
      # Abrir la comunicación con el dispositivo.
      print "device"
      device = evdev.Device(DEVICE_PATH)
      # Set some static state before entering the loop.
      print "read_fds"
```

```
      read_fds = [fifo, device.fd]
      print read_fds
      eventsize = evdev.Event.get_format_size()
      dato = ""
      try:
        print "while interno"
        # Se repite continuamente esperando por eventos de teclado o escáner.
        while True:
          # La clase select permite esperar por eventos relacionados a
archivos,
          # puertos, sockets, entre otros. Solo nos interesa la lectura.
          read, write, error = select.select(read_fds,[],[])
          if device.fd in read:
            # Nuevo evento de teclado o escáner.
            data = os.read(device.fd, eventsize)
            event = evdev.Event(unpack=data)
            # Si no hay evento, continúa.
            if event is None:
              print "No evento"
              continue
            command = None
            # Dentro de la descripción del evento, buscar el tipo EV KEY, que
implica
            # presión de una tecla. También existe la liberación de la tecla.
            if event.type == 'EV_KEY' and event.code.startswith("KEY") and
event.value == 1:
            # Los eventos de teclado inician con KEY_.
            # En el caso de los números inician con KEY_ si son realizados
con el
            # teclado alfanumérico.
            # Comienzan con KEY_KP si son realizados desde el teclado
numérico.
            # Si es tomado desde un escáner de códigos de barras como el que
se
            # está utilizando, los números los interpreta como si fueran
realizados
            # con el teclado numérico (KEY KP)
              command = event.code
              command = command.replace("KEY_KP","")
              command = command.replace("KEY_","")
              longitud = len(command)
              # Para hacerlo congruente con el teclado de membrana del
arduino.
              # Se sustituye el backspace por asterisco y el enter por un un
numeral.
              if longitud > 1:
                if command == 'BACKSPACE':
                  command = '*'
                elif command == 'ENTER':
                  command = '#'
                else:
                  command = None
            # Cuando el evento es enviado desde un scaner, si el código es de
10 dígitos
            # o más, hay problemas con el bridge al enviarlo dígito por
dígito.
```

```
            # Por lo tanto se almacenan los dígitos hasta encontrar un ENTER
y enviar
            # todos los dígitos de una sola vez hacia el sketch.
            if command is not None:
              dato = dato + command
              if command == "#":
                  bc.put('to arduino', dato)
                  print "enviar: " + dato
                  dato = ""
      finally:
        # En caso de fallo, cerrar el hilo del sistema que lee el FIFO.
        os.close(fifo)
        os.remove(FIFO_PATH)
    except:
      e = sys.exc_info()[0]
      print e
      traceback.print_exc()
    finally:
      print "No device"
      traceback.print_exc()
```

11.4 Firmware del prototipo

```
/*
Sketch hace uso de:
- Arduino Yun
- teclado matricial de 12 teclas.
- Display LCD de 16x2
- Conexión a Servidor web glassfish via protocolo REST.

Proceso:
A. Iniciar OF: empleado + op
B. Pausar OP: empleado
C. Registrar avance: empleado + op + cantidad
D. Ingreso: empleado + op + articulo + cantidad

Detalles:
Punto de Control: Accesorios...Definir punto en /root/config/puntoControl.txt
Multi-empleado: SI
Multi-OP:SI
Varias OP simultaneas por empleado: NO
Proceso: Varios, definidos en /root/config/procesos.txt
Servidor: Definido en /root/config/server.txt
*/

//Se incluye la libreria Keypad y liquidCrystal
#include "Keypad.h"
#include "LiquidCrystal.h"
#include "Bridge.h"
#include "HttpClient.h"
#include "Process.h"

//////////////////Teclado//////////////////////
const byte FILAS = 4;
const byte COLUM = 4;

char teclas[FILAS][COLUM] = {
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};

//arreglo con referencia a pines conectados a las filas del teclado.
byte pinesFilas[FILAS] = {10, 9, 8, 7};
//arreglo con referencia a pines conectados a las columnas del teclado.
byte pinesColumnas[COLUM] = {A0,A1,A2,A3};

Keypad miTeclado = Keypad(makeKeymap(teclas), pinesFilas, pinesColumnas,
FILAS, COLUM);


//Configuración de LCD
//LiquidCrystal lcd(5, 4, 3, 2, 1, 0);
LiquidCrystal lcd(12,11, 5, 4, 3, 2);
```

```
String numOp;
String empleado;
String articulo;
String cantidad;
String texto;
String cadenaXML;
String correcto;
String proceso;
boolean terminado=true; //Indica si el proceso finalizó o requiere más
parámetros.
String XValue = "X";
String modo; //E=empleado, O=OP.
String accion; //A=Iniciar OP,B=Pausar Trabajo, C=Preingreso, D=Trabajo
Actual.
int intento = 0;
String RUTA = "/root/comandos/";
String CONFIG = "/root/config/";
String puntoControl;
char D12value[15];

long checkPointEspera = 0;
long checkPointEsperaAnt = 0;
const long lapsoEspera = 1000L * 120L;

void setup() {

  //Configurando pantalla LCD
  lcd.begin(16,2);

  lcd.setCursor(0, 0);
  lcd.print(F("BIENVENIDO UDB"));

  Bridge.begin();
  Console.begin();
  printGeneric(F("inoSeguimientoUDB1.0"),0,0);
  delay(1000);

  while (cadenaXML != F("CONECTADO")) {
    intento += 1;
    printGeneric(F("Conectando..."),0,0);
    if (intento == 1) {
      delay(2000);
    } else {
      delay(5000);
    }
    getURLServer();
    getNumPuntoControl();
    pingServer();

    printGeneric(cadenaXML,0,0);
    delay(1000);

    if (cadenaXML != F("CONECTADO")) {
      printGeneric(F("SIN CONEXIÓN"),0,0);
      delay(3000);
    }
  }
```

```
  //Obteniendo dirección IP.
  getIP();

  //Iniciando funcione
  modo = "E";
  accion = "A";
  printText("");
  printFreeRAM();

}

void loop() {

  checkPointEspera = millis();

  char tecla;
  String vacio;
  char vacio2;

  bridgeGet();

  String caracter = String(D12value);

  if (caracter == XValue && caracter) {
    tecla = miTeclado.getKey(); //Si no presiona la tecla, la variable queda
vacÃa.
  } else {
    if (caracter.length() == 1) {
      tecla = caracter.charAt(0);
    } else {
      tecla = '#';
      texto = caracter;
      texto.replace("#","");
      texto.replace(" ","");
    }
    bridgePut();
    printText("");
    Console.println(texto);
  }

  //verificar que tecla presiono
  if (tecla == 'A' || tecla == 'B' || tecla == 'C' || tecla == 'D') {
    accion = String(tecla);
    texto = vacio;
    tecla = vacio2;
    modo = "E";
    printText("");
    checkPointEspera = millis();
    checkPointEsperaAnt = checkPointEspera;
  }

  if (tecla == '*') {
    int tamanio = texto.length();
    texto = texto.substring(0, tamanio -1);
    tecla = vacio2;
    printText(texto);
```

```
}

// Interpreta la tecla numeral como "Enter"
if (tecla == '#') {

  if (texto.length() > 0) {
    terminado = false;

    if (modo == "E") {
       empleado = texto;
    } else if (modo == "O") {
       numOp = texto;
    } else if (modo == "A") {
       articulo = texto;
    } else if (modo == "C") {
       cantidad = texto;
    } else if (modo == "P") {
       proceso = texto;
    }

    ejecutar();
    if (cadenaXML.length()>0) {
      printGeneric(cadenaXML,0,0);
      if (modo == "C" || terminado) {
         delay(2000);
      } else {
        delay(1000);
      }
    } else {
      printGeneric(F("ERROR SIN RESPUESTA"),0,0);
      delay(4000);
    }

    if (terminado) {
      trabajoActualPc();
      printGeneric(cadenaXML,0,0);
    } else {
      printText("");
    }
  }

  cadenaXML = "";
  texto = vacio;
  tecla =vacio2;
  printFreeRAM();

}

if (tecla) {
  texto = texto + tecla;
  printText(texto);
  checkPointEsperaAnt = checkPointEspera;
}

long diferenciaEspera = checkPointEspera - checkPointEsperaAnt;
if (diferenciaEspera > lapsoEspera) {
   checkPointEsperaAnt = checkPointEspera;
```

```
      notificarNuevaOf();
      if (correcto == "0"){
         trabajoActualPc();
      }
      printGeneric(cadenaXML,0,0);
      modo = "E";
   }

}

void bridgeGet() {
  Bridge.get("to_arduino", D12value, sizeof(D12value));

}

void bridgePut() {
  Bridge.put("to_arduino",XValue);
}


////////////////// BEGIN FLUJO DE OPCIONES ////////////////////////////
void ejecutar() {
  printGeneric(F("PROCESANDO......"),0,0);
  correcto = "1";

   if (accion == "A") { //Iniciar OP.
       if (modo == "O") {
           iniciarOf();

           if (correcto=="1") {
              modo = "E";
              terminado = true;
           }

       } else if (modo == "E") {
           validarEmpleado();

           if (correcto=="1") {
              modo = "O";
           }
       }  // if (modo)
   } else if (accion == "B") { //Pausar OP
       pausarOf();
       modo = "E";

       if (correcto=="1") {
        terminado = true;
       }

   } else if (accion == "C") {//Terminar

       if (modo == "E") {
           validarEmpleado();

           if (correcto=="1") {
              modo = "O";
           }
```

```
        } else if (modo == "O") {
            terminarProceso();

            if (correcto=="1") {
               modo = "E";
               terminado = true;
            }
        }

    } else if (accion == "D") {//Avance

        if (modo == "E") {
            validarEmpleado();

            if (correcto=="1") {
               modo = "O";
            }
        } else if (modo == "O") {
            validarOpAvance();

            if (correcto=="1") {
               modo = "A";
            }
        } else if (modo == "A") {
            validarArticulo();

            if (correcto=="1") {
               modo = "C";
            }
        } else if (modo == "C") {
           ingreso();

            if (correcto=="1") {
               modo = "A";
            }
        } //if modo


    }
    printFreeRAM();
}

////////////////// BEGIN FLUJO DE OPCIONES ////////////////////////////

//////////////////BEGIN BUSSINESS LOGIC//////////////////////////
void validarEmpleado() {

    Console.println(RUTA + "validarEmpleado.sh" + " " + empleado);
    Process p;        // Crea proceso en variable P
    p.begin("ash");
    p.addParameter(RUTA + "validarEmpleado.sh");
    p.addParameter(empleado);
    p.run();
    getMessage(p);
}

void iniciarOf() {
```

```
    Console.println(RUTA + "iniciarOf.sh" + " " + empleado + " " + numOp);
    Process p;        // Crea proceso en variable P
    p.begin("ash");
    p.addParameter(RUTA + "iniciarOf.sh");
    p.addParameter(empleado);
    p.addParameter(numOp);
    p.run();
    getMessage(p);
}

void pausarOf() {

    Console.println(RUTA + "pausarOf.sh" + " " + empleado);
    Process p;        // Crea proceso en variable P
    p.begin("ash");
    p.addParameter(RUTA + "pausarOf.sh");
    p.addParameter(empleado);
    p.run();
    getMessage(p);
}

void terminarProceso() {

    Console.println(RUTA + "terminarProceso.sh" + " " + empleado + " " +
numOp);
    Process p;        // Crea proceso en variable P
    p.begin("ash");
    p.addParameter(RUTA + "terminarProceso.sh");
    p.addParameter(empleado);
    p.addParameter(numOp);
    p.run();
    getMessage(p);
}

void validarOpAvance() {

    Console.println(RUTA + "validarOfAvance.sh" + " " + empleado + " " +
numOp);
    Process p;        // Crea proceso en variable P
    p.begin("ash");
    p.addParameter(RUTA + "validarOfAvance.sh");
    p.addParameter(empleado);
    p.addParameter(numOp);

    p.run();
    getMessage(p);
}

void validarArticulo() {

    Console.println(RUTA + "validarArticulo.sh" + " " + empleado + " " +
numOp + " " + articulo);
    Process p;        // Crea proceso en variable P
    p.begin("ash");
    p.addParameter(RUTA + "validarArticulo.sh");
    p.addParameter(empleado);
```

```
    p.addParameter(numOp);
    p.addParameter(articulo);

    p.run();
    getMessage(p);
}

void ingreso() {
    Console.println(RUTA + "ingreso.sh" + empleado + "-" + numOp + "-" +
articulo + "-" + cantidad);
    Process p;        // Crea proceso en variable P
    p.begin("ash");
    p.addParameter(RUTA + "ingreso.sh");
    p.addParameter(empleado);
    p.addParameter(numOp);
    p.addParameter(articulo);
    p.addParameter(cantidad);

    p.run();
    getMessage(p);
}

void trabajoActualPc() {
    Console.println(RUTA + "consultarTrabajoPunto.sh");
    Process p;        // Crea proceso en variable P
    p.begin("ash");
    p.addParameter(RUTA + "consultarTrabajoPunto.sh");
    p.run();
    getMessage(p);
}

void notificarNuevaOf() {
    Process p;
    p.begin("ash");
    p.addParameter(RUTA + "notificarNuevaOf.sh");
    p.run();
    getMessage(p);
}


//////////////////END BUSSINESS LOGIC//////////////////////////

/////////////// BEGIN FUNCIONES AUXILIARES /////////////////

void pingServer() {

    Console.println(RUTA + "pingServer.sh");
    Process p;        // Crea proceso en variable P
    p.begin("ash");
    p.addParameter(RUTA + "pingServer.sh");
    p.run();

    getMessage(p);
}

void getURLServer() {
  Process p;
```

```
  p.begin("cat");
  p.addParameter(RUTA +"server.txt");
  p.run();
  cadenaXML = "";
  while (p.available()) {
    char c = p.read();
    cadenaXML += c;
  }

 cadenaXML.trim();
 printGeneric(cadenaXML, 0, 0);
 delay(500);
}

void getIP() {
  Process p;
  p.begin("ash");
  p.addParameter(F("/root/config/getIp.bat"));
  p.run();

  cadenaXML = "";

  while (p.available()) {
    char c = p.read();
    cadenaXML += c;
  }

  cadenaXML.replace("addr:","");
  printGeneric("IP:" + cadenaXML, 0, 0);
  delay(1000);

}

void getNumPuntoControl() {
  Process p;
  p.begin("cat");
  p.addParameter(CONFIG + "puntoControl.txt");
  p.run();
  puntoControl = "";
  while (p.available()) {
    char c = p.read();
    puntoControl += c;
  }
 puntoControl.trim();
 printGeneric("PUNTO:" + puntoControl, 0, 0);
 delay(500);
}

void getMessage(Process p){

  Console.println(F("Getting message"));
  cadenaXML = "";
  boolean concatenar = false;
  boolean returnFound = false;

  while (p.available()) {
```

```
      char c = p.read();

    //Nota: Dado que el arduino es limitado en recursos de memoria sobre todo
al manejar la clase String
    //no se almacenará toda la cadena XML enviada por el servidor, sino se
buscará la etiqueta <return>
    //y desde ahi se almacenará hasta que se encuentre la etiqueta de cierre
</return>

        if (returnFound) {
       if (cadenaXML.indexOf("</respuesta>") == -1){
       cadenaXML += c;
       } // (cadenaXML.indexOf("</return>")
    }//(returnFound)

    if (c == '<') {
      concatenar=true;
    }

    if (concatenar && !returnFound) {
      cadenaXML += c;

      if (cadenaXML.length() == 11) {
      if (cadenaXML != "<respuesta>") {
        cadenaXML = "";
        concatenar = false;
      } else {
        returnFound = true;
      }
      }
    } //(concatenar && !returnFound)

     } //while (p.available())

     cadenaXML.replace("<respuesta>","");
     cadenaXML.replace("</respuesta>","");
     correcto = String(cadenaXML.charAt(0));
     cadenaXML = cadenaXML.substring(2);
     Console.print(F("Respuesta: "));
     Console.println(cadenaXML);
}

void printText(String texto) {
  lcd.clear();
  lcd.setCursor(0, 0);

  if (modo == "E") {
    lcd.print("EMPLEADO:" + texto);
  } else if (modo == "O") {
    lcd.print("NUM OP:" + texto);
  } else if (modo == "A") {
    lcd.print("ARTIC:" + texto);
  } else if (modo == "C") {
    lcd.print("CANTIDAD:" + texto);
  }

  lcd.setCursor(0, 1);
```

```
  if (accion == "A") {
    lcd.print(F("INICIAR OP      "));
  } else if (accion == "B") {
    lcd.print(F("PAUSAR/REANUDAR "));
  } else if (accion == "C") {
    lcd.print(F("TERMINAR PROCESO"));
  } else if (accion == "D") {
    lcd.print(F("INGRESO A BODEGA"));
  }
}

void printGeneric(String texto, int fila, int columna) {
  lcd.clear();
  lcd.setCursor(fila, columna);
  lcd.print(texto);

  if (texto.length() > 16) {
    lcd.setCursor(0,1);
    lcd.print(texto.substring(16));
  }
}

///////////////// END FUNCIONES AUXILIARES /////////////////
int freeRam ()
{
  extern int __heap_start, *__brkval;
  int v;
  return (int) &v - (__brkval == 0 ? (int) &__heap_start : (int) __brkval);
}

void printFreeRAM(){
  Console.println("Disponible:");
  Console.print(freeRam());
}
```

11.5 Historial de pruebas básicas

ID	Fecha	Hecho por	Descripción	Categoría	Éxitos	Fallos	Siguiente Prueba	Anterior
1	27/03/2016	LA	Conectar el Arduino Yun y cargar el panel web de administración. Se sigue la guía encontrada en https://www.arduino.cc/en/Guide/ArduinoYun. Componentes: Arduino Yun, Cable micro USB y navegador.	SOFTWARE	Después de conectarse a la red inalámbrica Arduino Yun-XXXX; se pudo ingresar al panel web a través de la dirección IP por defecto (192.168.240.1). La contraseña por defecto es "Arduino".	No se pudo acceder, según lo especifica la guía con http://arduino.local		
2	27/03/2016	LA	Actualizar el sistema operativo a la versión más reciente, según lo indica en la documentación: http://arduino.cc/en/Tutorial/YunSysupgrade. Se descargó la imagen y se copió en una tarjeta micro SD de 4GB. Luego se accedió al panel web donde solicitó la actualización.	SOFTWARE	Se realizó correctamente la actualización del Sistema Operativo OpenWrt-Yun.	Ninguno		

ID	Fecha	Hecho por	Descripción	Categoría	Éxitos	Fallos	Siguiente Prueba	Anterior
3	27/03/2016	LA	Expandir la memoria interna con un micro SD. Se siguió el proceso encontrado en: http://arduino.cc/en/Tutorial/ExpandingYunDiskSpace. Se cargó en la memoria del Arduino el sketch "YunDiskSpaceExpander".	SOFTWARE	El sketch se cargó e inició solicitando la confirmación, pero no finalizó.	El proceso se detuvo con error: Downloading http://downloads.arduino.cc/openwrtyun/1/packages/Packages.gz. Updated list of available packages in /var/opkg-lists/attitude_adjustment. Downloading http://downloads.arduino.cc/openwrtyun/1/packages/Packages.sig. Signature check failed. RemovewrongSignature file.	Verificar funcionamiento del gestor de paquetes del SO del Yun	
4	27/03/2016	LA	Comentar la verificación de la firma de los paquetes descargados. Ver #3.	SOFTWARE	El proceso de expansión de memoria se realizó exitosamente.			3

ID	Fecha	Hecho por	Descripción	Categoría	Éxitos	Fallos	Siguiente Prueba	Anterior
5	28/03/2016	LA	Conectar a red inalámbrica. La conexión se realiza a una red empresarial con una antena Aruba Networks con seguridad WPA2-Enterprise. La conexión se realiza desde la consola web del Arduino Yun en la sección "Wireless", escaneando las redes disponibles.	RED	El Arduino identificó la red inalámbrica a conectarse.	No se logró la conexión debido a que el SO no soporta de fábrica la autenticación WPA2 Entreprise.	De acuerdo a la documentación el paquete encargado de la conexión es el wpad. Sin embargo, Linino trae instalado wpad-mini, una versión light del paquete. Es necesario realizar la instalación del paquete completo.	
6	28/03/2016	LA	Instalación del paquete wpad. Es necesario actualizar el gestor de paquetes OPKG para instalar. Se utilizó la siguiente secuencia de comandos: opkgupdate opkgremovewpad-mini opkginstallwpad	RED	El Arduino logró realizar momentáneamente la conexión y luego se perdió dicha conexión.	1. Desde la consola de administración de la antena se observa el dispositivo intentando conectarse, pero siendo denegada la conexión. 2. Cuando se aleja de la antena, se observa reinicio del Linino.	Consultar el log del SO para detectar causas de reinicios.	5
7	30/03/2016	LA	Búsqueda de logs de errores para determinar causas de reinicios	SOFTWARE	Se encuentra que al reiniciarse las conexiones de red vuelven a las configuraciones originales para utilizar el Arduino como accesspoint, en lugar de dispositivo cliente.	No se identificó causa de reinicios.	Consultar los scripts que corren en el inicio de SO.	6

ID	Fecha	Hecho por	Descripción	Categoría	Éxitos	Fallos	Siguiente Prueba	Anterior
8	01/04/2016	LA	Dar seguimiento a los scripts de inicio. Por default la versión de Linino busca en el archivo /etc/rc.local	RED	El Arduino Yun tiene un script que corre en backgroud con el cual verifica si hay conectividad Wifi. En el caso, por ejemplo, que se configure la red inalámbrica propia y al encender, este no la encuentra, este script procede a apagar el SO del Arduino y luego a setear la configuración por defecto, es decir, regresa la configuración inalámbrica a la red Arduino Yun- XXXXX. Se comentó la línea "wifi-live-or-reset" del archivo /etc/rc.local	No se conecta a la red inalámbrica de manera estable. La antena sigue mostrando problemas de autenticación.		
9	02/04/2016	LA	Se observa que la hora del sistema está desactualizada. Se procede a actualizar y luego se procede a la conexión	RED	El Yun se conectó exitosamente a la red WPA2-Enterprise			

11.6 Historial de pruebas de armado del prototipo

ID	Fecha	Hecho por	Descripción	Categoría	Éxitos	Fallos	Siguiente Prueba	Anterior
1	02/04/2016	LA	Conexión de Arduino con pantalla de cristal líquido. Componentes: - Arduino Yun. - Pantalla LCD 16x2 (compatible con el driver Hitachi HD44780). - Potenciómetro de 10 KOhm. - Cables jumper macho/hembra. Para mayor información en la conexión de la pantalla, ver la guía "Configuración de hardware prototipo Arduino Yun". Para la prueba, se carga el script de prueba encontrado en: https://www.arduino.cc/en/Tutorial/HelloWorld	HARDWARE	Se realiza la conexión de manera exitosa, desplegando correctamente los datos que genera el sketch de prueba.	Debido a que se necesitan 3 cables conectados a 5v, 3 cables a GND y uno a la resistencia del potenciómetro, se hace difícil utilizar el potenciómetro provisto de fábrica con la pantalla, el cual es plástico.	Utilizar un potenciómetro más resistente.	
2	03/04/2016	LA	Conexión de Arduino con teclado de membrana (4x4). Componentes: - Arduino Yun. - Teclado de membrana (4x4). - Cables jumper macho/macho. Para mayor información en la conexión de teclado, ver la guía "Configuración de hardware prototipo Arduino Yun".	HARDWARE	Se realiza la conexión del teclado. Todas las teclas funcionan	La tercera columna [3, 6, 9, #], se encuentra intercambiada con la cuarta columna [A, B, C, D].	Verificar la especificación del teclado de membrana para realizar la conexión de manera adecuada.	

ID	Fecha	Hecho por	Descripción	Categoría	Éxitos	Fallos	Siguiente Prueba	Anterior
3	03/04/2016	LA	Se conecta el Arduino con el teclado, siguiendo esta configuración: K1:D10, K2:D9, K3:D8, K4:D7, K5:A0, K6:A1, K7:A2, K8:A3. Siendo Kn, el número de pin de entrada del teclado de membrana; y, An y Dn el número de pin digital o analógico al que se conectó, respectivamente.	HARDWARE	Teclado de membrana funcionando correctamente.			
4	03/04/2016	LA	Conexión de lector de barras USB al Arduino Yun. Componentes: - Arduino Yun. - Escáner USB (Más información en ficha técnica de Arduino Yun).	HARDWARE		El sketch no es capaz de reconocer los datos tomados con el escáner. Al realizar una lectura del código de barras, el escáner queda realizando un sonido continuo. De acuerdo a la hoja técnica esto sucede cuando no puede ser reconocido por la computadora a la que está conectada.	Verificar la conexión de dispositivos USB desde el SO del Arduino Yun.	

ID	Fecha	Hecho por	Descripción	Categoría	Éxitos	Fallos	Siguiente Prueba	Anterior
5	04/04/2016	LA	Conectarse vía SSH al Linino y verificar si el Yun reconoce dispositivos USB. Se conecta un dispositivo USB, un teclado en este caso y se verifica si es reconocido por el sistema operativo, a través del comando "lsusb"	HARDWARE		El teclado USB no es reconocido dentro de la lista de dispositivos USB conectados. Si se usa el comando dmesg (para mensajes de diagnóstico), se observa que logra enumerar el dispositivo	Investigar qué paquetes faltan en el SO para que el dispositivo USB sea reconocido correctamente	
6	05/04/2016	LA	De acuerdo con lo indicado en el foro de Arduino: http://forum.arduino.cc/index.php?topic=207069.0 Es necesario instalar los paquetes para añadir soporte a dispositivo HID. https://wiki.openwrt.org/doc/howto/usb.essentials. Kmod-input-evdev: Módulo del kernel para soportar eventos de dispositivos de entrada. Kmod-hid: módulo del kernel para dispositivos de Interfaz Humana (HID).	HARDWARE	Luego de instalar los paquetes se puede observar una nueva entrada en la lista de dispositivos. Se puede corroborar navegando a la ruta /dev/input. Al listar (ls) se observa event0 y event1. Con el comando cat event1 | hexdump se puede ver el equivalente en hexadecimal de las teclas presionadas.	Aun no es posible reconocer las entradas generadas por el teclado USB desde el sketch. Además es necesario traducir los datos hexadecimales a caracteres alfanuméricos.	Convertir a caracteres alfanuméricos las entradas del teclado.	5

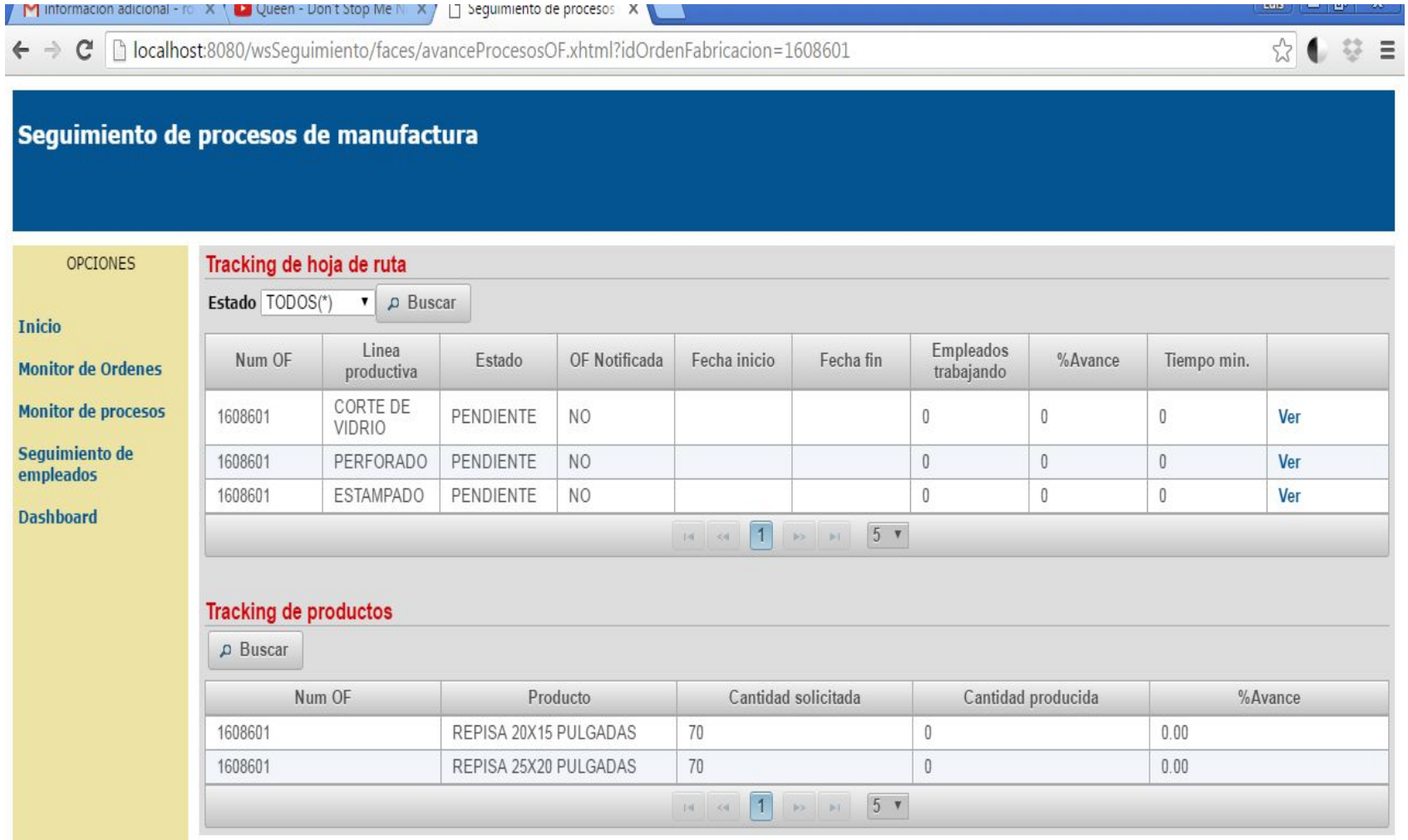
ID	Fecha	Hecho por	Descripción	Categoría	Éxitos	Fallos	Siguiente Prueba	Anterior
7	07/04/2016	LA	De acuerdo con: https://www.kernel.org/doc/Documentation/input/input.txt. El módulo evdev, es un conjunto de drivers, diseñados para soportar todos los dispositivos de entrada en Linux. Desde la consola de Linino se puede ejecutar el script evdev.py indicando el listener de eventos a utilizar (event1, según la prueba 6).	INPUT	Al utilizar el script para eventos se puede observar las teclas presionadas en el teclado USB. El módulo evdev retorna los siguientes datos de los dispositivos de entrada: - time: devuelve la fecha y hora del evento. - code: Código de evento, como KEY_KP1 para la tecla uno de un teclado numérico, entre otros. - Value: Estado del evento, para el teclado; 1, tecla presionada; 0, tecla liberada; 2, auto repetición.		Tomar el valor de las teclas presionadas y enviarlas por el bridge para ser utilizadas en el sketch.	6

ID	Fecha	Hecho por	Descripción	Categoría	Éxitos	Fallos	Siguiente Prueba	Anterior
8	11/04/2016	LA	Se utiliza un script de Python, basado en https://github.com/tdicola/EvilMousePrank; para enviar las teclas presionadas en el teclado a través del Bridge de Arduino Yun. Las teclas presionadas son enviadas al sketch de ejemplo para verificar la interacción.	INPUT	El sketch reconoce las teclas presionadas.	El tiempo de respuesta entre tecla presionada es elevado. Tarda un aproximado de 1 segundo para que el sketch reconozca cada tecla. Además, se observa que al reinicio del Arduino, el display muestra caracteres especiales. Esto podría ser por utilizar comunicación serial. Ya que el Yun despliega vía Serial las secuencias de arranque del SO.	Disminuir el tiempo de respuesta del sketch al ingresar datos desde teclado. Utilizar la comunicación vía Bridge en lugar de utilizar la comunicación Serial para evitar interferencia.	7
9	12/04/2016	LA	Utilizar el módulo de PythonBridgeClient para enviar los datos digitados por medio del teclado para el sketch del Arduino. Esta clase sustituirá la comunicación Serial.	INPUT	El sketch reconoce las teclas presionadas y ya no despliega caracteres especiales durante el inicio del sistema operativo.	El tiempo de respuesta entre las teclas presionadas sigue siendo elevado. En ocasiones la tecla digitada se repite, como si fuese digitada varias veces.	Probar la comunicación vía Bridge para determinar la causa de la repetición de los caracteres digitados.	8

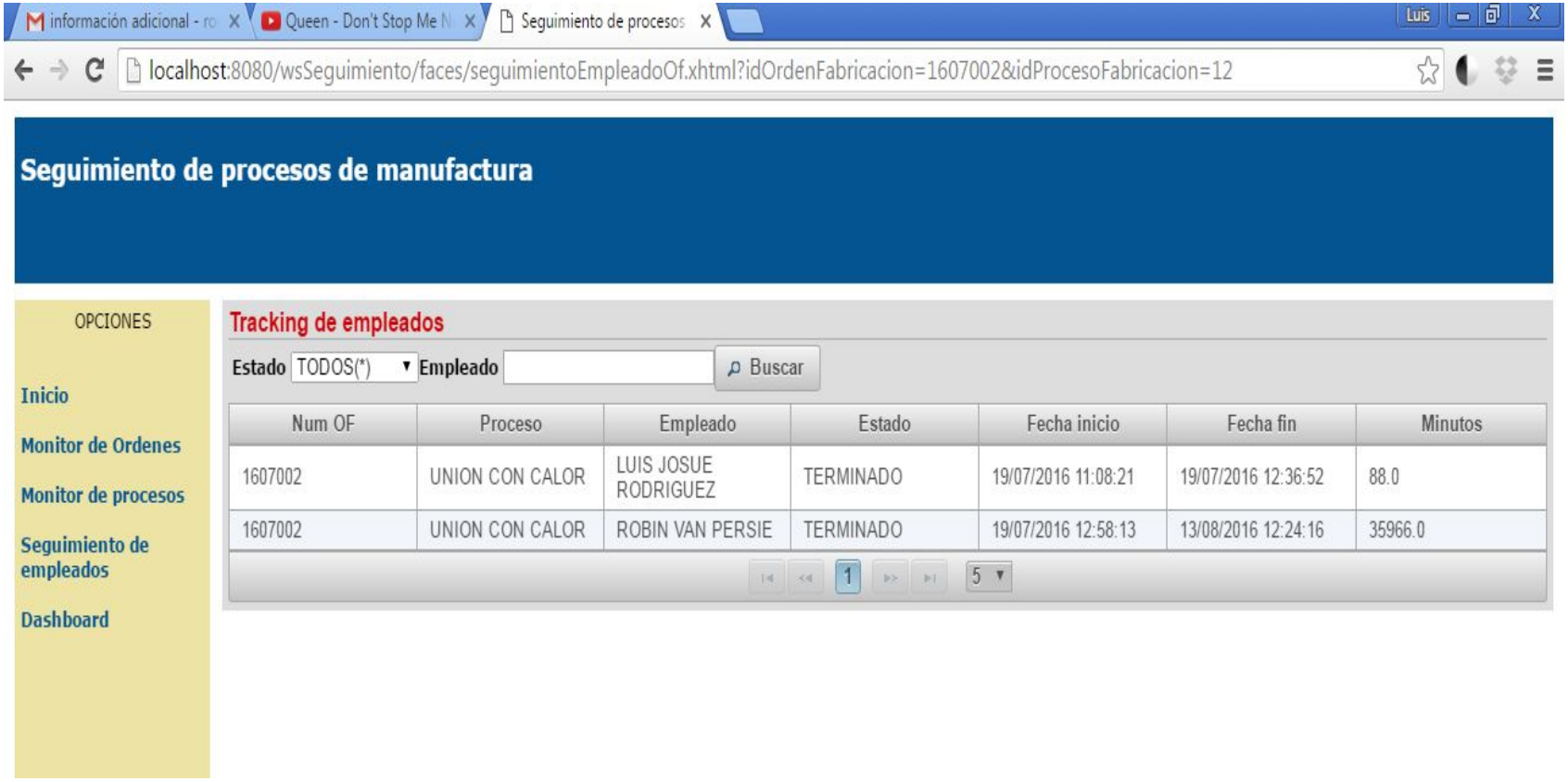
ID	Fecha	Hecho por	Descripción	Categoría	Éxitos	Fallos	Siguiente Prueba	Anterior
10	12/04/2016	LA	Se envían datos desde el teclado y se observa que el dato enviado por el bridge permanece disponible en el buffer hasta que detecta un evento. Para evitar que eso suceda, se limpia el valor de la variable capturada en el sketch de nuevo hacia el Linino.	INPUT	El sketch reconoce las teclas presionadas. Además la velocidad de respuesta se mejoró y los caracteres presionados son reconocidos de manera inmediata en el sketch.			

11.7 Pantallas de Seguimiento (Órdenes de Fabricación en Proceso)

11.8 Pantallas de Seguimiento (Estado del avance de OF)



11.9 Pantallas de Seguimiento (Seguimiento de Empleados por OF)



11.10 Guía de armado de hardware del prototipo

CONFIGURACIÓN DE HARDWARE PROTOTIPO ARDUINO YUN

Introducción

El siguiente tutorial busca realizar paso a paso la instalación de hardware para tener un Arduino Yun completamente armado.

Producto final

Al final de este tutorial tendrás la capacidad de tener el Arduino Yun ensamblado y listo para ser utilizado en producción, asumiendo que seguiste la guía “Configuración de Software Arduino Yun”.

Requisitos y materiales

Los componentes necesarios para desarrollar el tutorial son:

COMPONENTES	
La placa ARDUINO YUN	**Carcasa para Arduino**
	

<table>
<tr><th colspan="2">COMPONENTES</th></tr>
<tr><td>Potenciómetro de 3 pines de 10 – 20 kilo ohmios</td><td>Kit de pantalla LCD + headers</td></tr>
<tr><td>  </td><td>  </td></tr>
<tr><td>Teclado de membrana para Arduino</td><td>Cables jumper macho/hembra y macho/macho para Arduino</td></tr>
<tr><td> 
</td><td>  </td></tr>
<tr><td>Accesorios para soldar (Cautín, estaño, base para soldar)</td><td></td></tr>
<tr><td>  </td><td></td></tr>
</table>

Procedimiento

Indicaciones generales

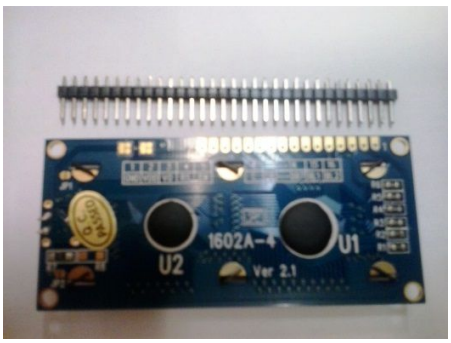
Según la experiencia anterior se puede resumir el armado del Arduino Yun en los siguientes:

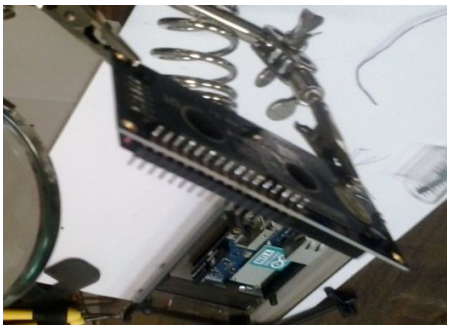
- Ensamble de la tapa inferior: incluye la tapa inferior de la carcasa más el Arduino Yun.
- Ensamble de la pantalla soldando el cableado y el potenciómetro.
- Ensamble del teclado.
- Atornillado final.

Paso 1: Ensamble de tapa inferior

Atornillar el Arduino Yun a la tapa inferior de la carcasa. Nota: Decida cuál será la inferior y la superior.

Paso 2: Soldar pines y pantalla





Soldar con cautín y estaño los pines que trae la pantalla a la pantalla LCD. Ya que el paquete de pines trae más de los necesarios, puede quebrar la tablilla para que quede la cantidad exacta. Tome en cuenta que debe colocar el lado más largo de los pines hacia el lado trasero de la pantalla para que luego se puedan soldar los cables jumper hembra.

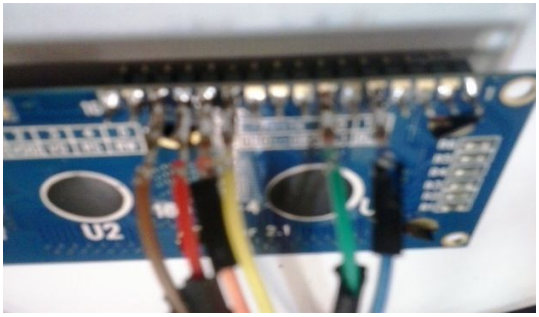
Paso 3: Soldar pines con cables jumper

Luego que haya quedado sólida la soldadura procedemos a soldar los pines desde la parte trasera a los cables jumper hembra/macho, de acuerdo al siguiente esquema:

Viendo la pantalla desde esta posición, donde el pin1 es el que estás más a la derecha y el pin 16 es el que está más a la izquierda.

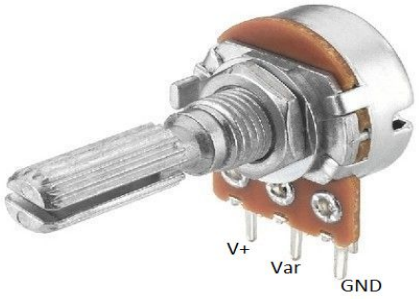
Haremos un paquete de 6 cables jumper macho-hembra y los soldaremos (soldado del conector hembra y quitándole el cobertor plástico) de la siguiente forma. Estos cables irán finalmente al Arduino Yun según un esquema que se mostrará posteriormente.

16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
		X	X	X	X					X		X			



Puede evitar que los cables choquen entre sí con cinta aislante o con el mismo empaque plástico que trae el conector hembra del cable jumper.

Paso 4: Soldar el potenciómetro



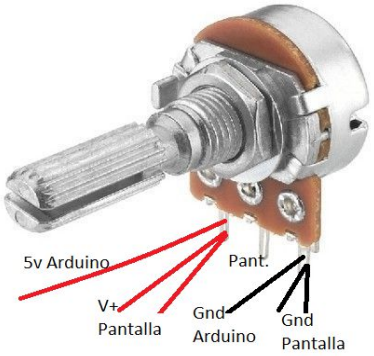
El potenciómetro es un dispositivo que nos permite regular la intensidad del brillo que tendrá el display LCD. Un potenciómetro básicamente es una resistencia variable por lo que nos permite amplificar o disminuir la cantidad de corriente que pasa al regulador de brillo de la pantalla. Nota: El kit de la LCD ya viene con un potenciómetro, sin embargo, es muy pequeño y difícil de ensamblar como se desea. Los potenciómetros pueden variar, pero el que estamos utilizando tiene el siguiente esquema.

V+: 5 Voltios al Arduino.

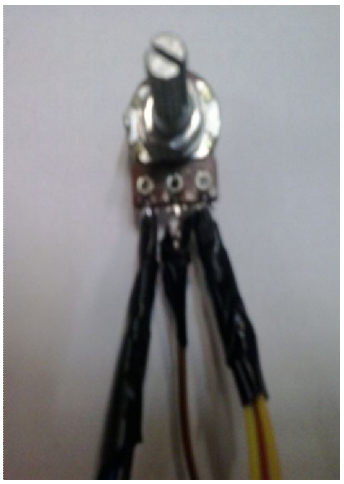
Var: Flujo de corriente variable del potenciómetro a la pantalla.

GND: A tierra del Arduino.

Con los datos anteriores hay que resaltar que tanto del pin V+ como del pin GND saldrá un cable jumper macho al pin 5V del Arduino y 2 a la pantalla, es decir, debe ser soldado de la siguiente forma.



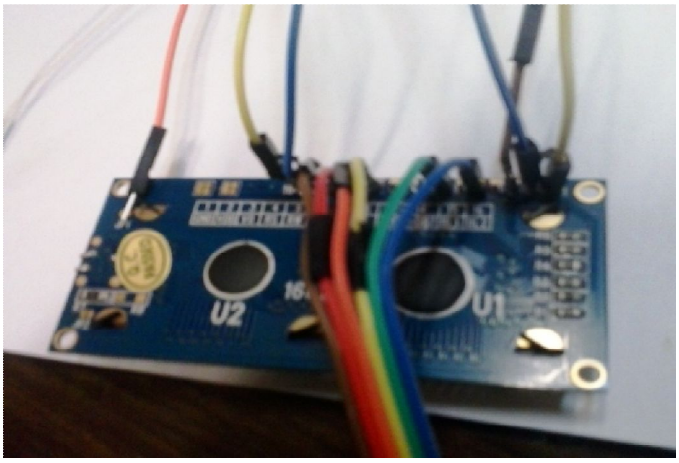
Hay que tener en cuenta que en el extremo de cable que NO va al potenciómetro, los que van al Arduino deben ser machos y los que van a la pantalla deben ser hembra. El potenciómetro ya soldado se verá de la siguiente forma:



Los cables que no van al Arduino, se soldarán en los pines del display LCD según este esquema:

16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
GND	V+										GND		VAR	V+	

La pantalla luego de ser soldada completamente, se verá así:



Paso 5: Ensamble la pantalla con la carcasa

Luego de tener soldado el potenciómetro a la pantalla, atorníllela a la parte frontal de la carcasa del Arduino Yun. La carcasa NO trae los tornillos adecuados.

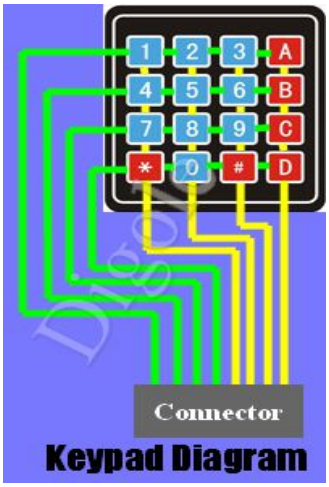
Nota: Es muy recomendable que después de terminar la labor de soldadura verifique con un multímetro que hay continuidad entre el punto de la pantalla LCD con el extremo macho de cada cable. Asimismo verificar que no se hayan unido puntos con el estaño porque alterará el funcionamiento de dicha pantalla.

Luego debe realizar el cableado de la pantalla hacia el Arduino Yun. Dado que usamos conectores macho-hembra, sólo será necesario introducir el conector macho en los pines correspondientes del Arduino, de acuerdo al siguiente esquema, el cual es continuación de los anteriores. La fila 1 (Pantalla) es un esquema de todos los pines de la pantalla LCD y a qué pin del Arduino (color amarillo); o, del potenciómetro (color celeste) deben conectarse. La fila 2 (Arduino) indica a que pin del Arduino irá conectado cada pin de la pantalla y la fila 3 (Potenciómetro) a qué pin del potenciómetro se conectará el pin de la pantalla. En blanco quedan los pines de la PANTALLA que no se conectan.

Pantalla	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1
Arduino			2	3	4	5					11		12			
Potenciómetro	GND	V+										GND		Var	V+	

Paso 6: Conectar teclado de membrana

El teclado de membrana es lo último que colocamos. Este no va soldado, ocupamos cables jumper tipo macho-macho para conectar tanto los extremos de la cincha en el teclado, como los pines del Arduino. Visto desde el frente, la conexión quedaría de la siguiente forma.



Keypad	1	2	3	4	5	6	7	8
Arduino	D10	D9	D8	D7	A0	A1	A2	A3

Paso 7: Armar arduino

Luego de realizar todas las conexiones, procedemos a atornillar la carcasa con todos sus componentes dentro. Los tornillos para cerrarlo vienen con la carcasa.