UNLaM

Sistemas Operativos Avanzados
Proyecto: SmartMailBox

Informe Final

Docentes:

- Waldo A. Valiente
- Mariano Volker
- Carnuccio, Esteban Andrés
- Gerardo Garcia

Integrantes:

- Celestino, Gustavo
- Gutiérrez, Ruben
- Montenegro, Diego
- Fontán, Andres
- Salmin, Silvio

Índice

Motivación del proyecto

¿No lo fastidia tener que ir al correo a buscar un paquete que compro por internet y que, como no había nadie en casa para recibirlo el cartero se lo llevo? Con SmarMailBox ya no tiene que haber alguien si o si en el hogar ni tampoco ir a buscar el paquete al correo.

SmartMailBox destaca por 5 características clave:

- **Paquetes esperados desde la app:** mediante una aplicación Mobile el cliente avisa al buzón que paquetes espera recibir
- **Almacenamiento y notificación:** SmartMailBox se encarga almacenar y notificar cuando llego el pedido esperado.
- **Siempre al tanto:** Se envían notificaciones al Cliente sobre el estado de su pedido en caso de que se trate de un pedido sensible a la temperatura.
- **Seguro:** SmartMailBox notificara alguna irregularidad con la apertura de la puerta o la desaparición de uno o más paquetes.
- **Identificación paquete:** Gracias al código y al peso del paquete SmartMailBox es capaz de indentificar que paquete llego y/o saco del buzón

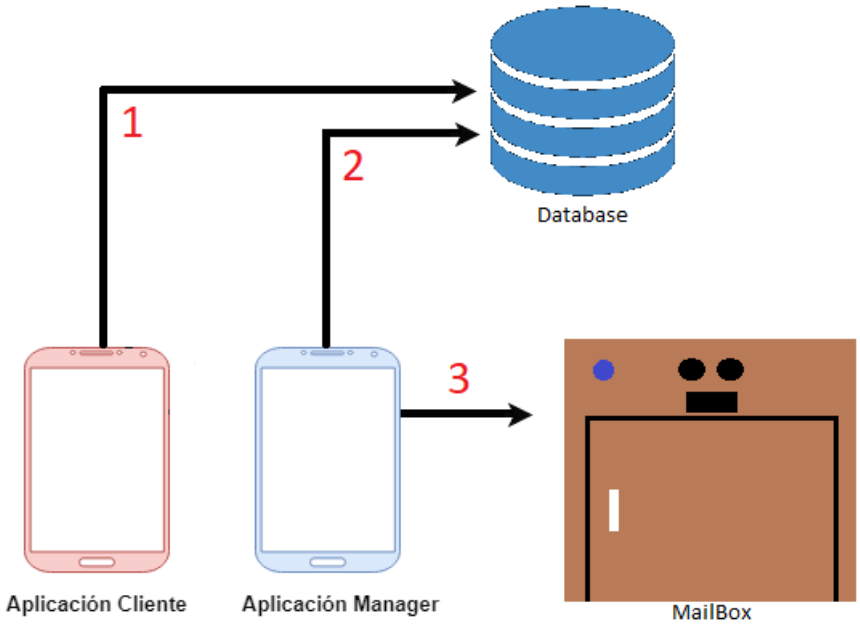
Estructura y funcionamiento general del proyecto

Estructura

Para cumplir con su objetivo, Proyecto SmartMailBox presenta 3 elementos principales:

- **Aplicación Cliente:** La cual debe ser instalada en un dispositivo móvil que disponga de conexión a internet para poder cargar en la base de datos pueda localizarlo, y acceso a conexión Wi-Fi para poder enviar su pedido.
- **Aplicación Manager:** La cual debe ser instalada en un dispositivo móvil que disponga de Bluetooth y conexión a internet para poder transmitir los datos necesarios desde el buzón a la base de datos y/o viceversa.
- **Buzon SmartMailBox**: El cual viene equipado con un módulo Bluetooth el cual utiliza para comunicarse con la **Aplicación Manager**.

Funcionamiento

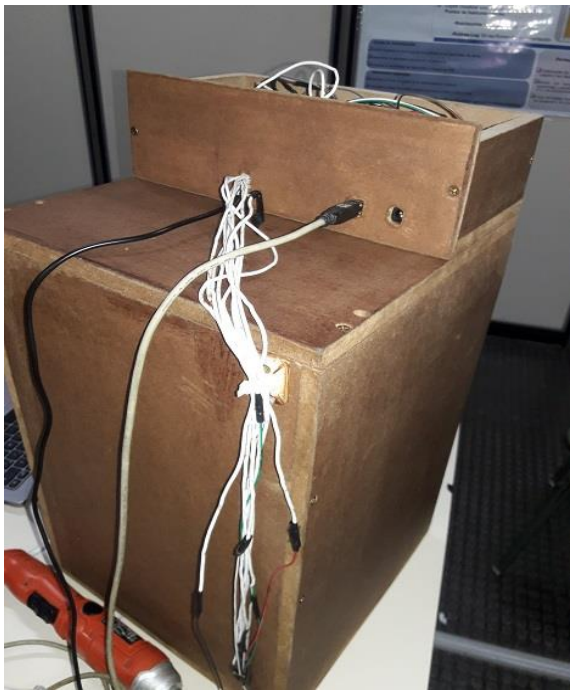


1. El Cliente, desde la Aplicación Cliente, deberá enviar el código del paquete que esté esperando recibir. Dichos pedidos son enviados a la Base de Datos.
2. Los códigos de los pedidos que lleguen a la base de datos luego serán cotejados con el código que escaneara la persona que traiga el paquete para dejarlo en el buzón.
3. El buzón SmartMailBox, activara el scanner de códigos una vez que detecte algo cerca, si se escanea el código de un paquete, envía vía bluetooth el código a la **Aplicación Manager**, la cual enviara por internet a la base de datos este mismo código, en la base de datos se realiza la cotejacion. En caso de ser un código que se estaba esperando, se procede a deshabilitar la traba de la puerta, caso contrario, se encenderá el aviso luminoso y llegara a la **Aplicación Cliente** la notificación correspondiente, tanto si se depositó el paquete esperado o si escanearon un código correcto.

Proyecto: Sistema Embebido

Concepto

SmartMailBox surge de la necesidad de no tener que estar en tu casa esperando por la persona que te tiene que entregar algo que ordenaste por internet o por otro medio, otra necesidad de uso surge cuando se vive en un edificio y no te encontras en condiciones de bajar a buscar el paquete o sencillamente no te encontras en tu hogar o te encontras durmiendo, sabes que te lo pueden dejar en el buzón de forma facil y segura.

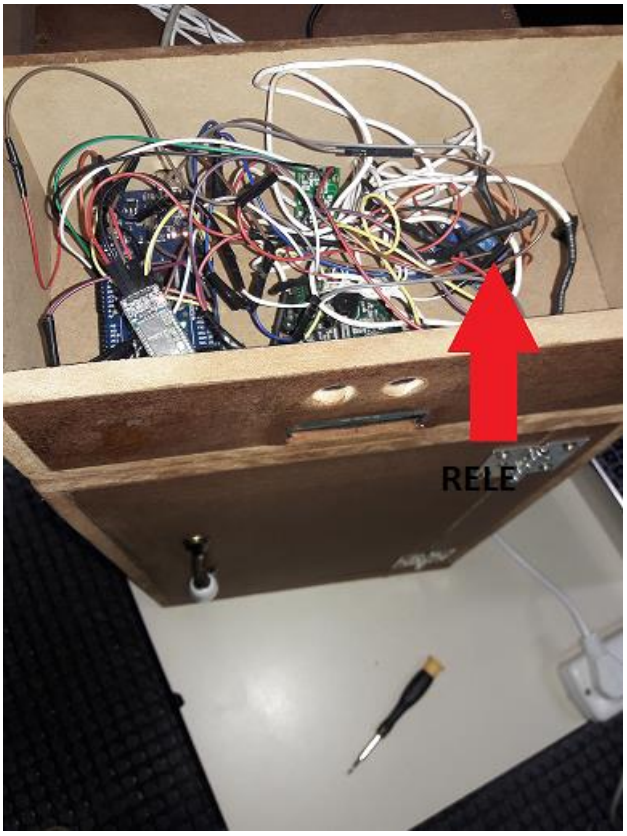


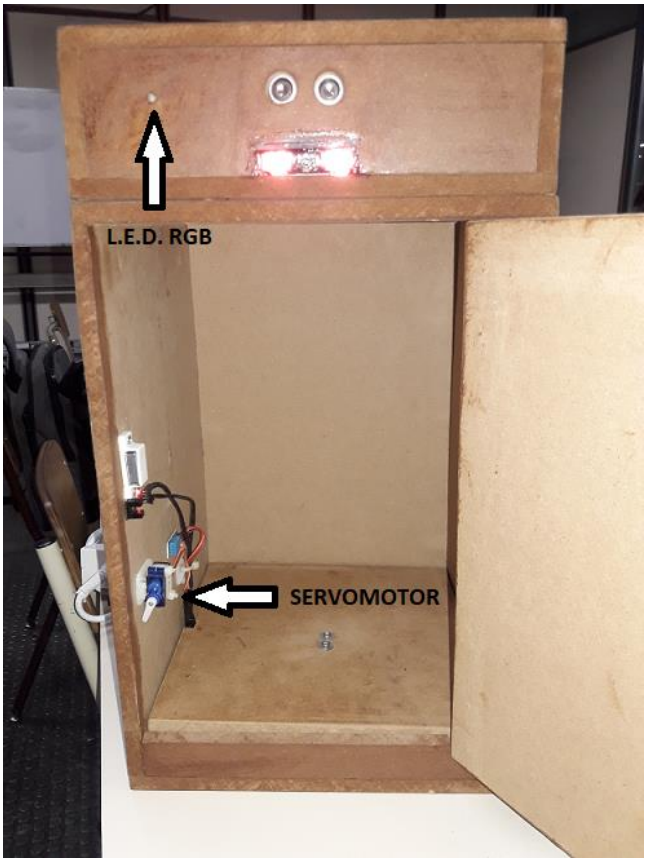
Sensores

Los sensores de SmartMailBox detectan cuando se tiene un objeto frente a su scanner de códigos, cuando se deposito o quita algo del buzón y si esta modificando la temperatura ambiente interna del buzón.
Sus Sensores son: Un sensor bluetooth a través del cual se realiza el envio del código a la base de datos y las respectivas notificaciones a la **Aplicación Manager**.
Ademas, cuenta con una balanza interna la cual permite detectar si realmente se deposito algo adentro del buzón, si abrieron la puerta escaneando el código correcto pero no dejaron nada o si disminuyo el peso cuando había un paquete dentro y no fuimos nosotros quien lo quito. En estos casos se enviara a **la Aplicación Cliente** la notificación correspondiente. Un sensor optico de barrera el cual indica si la puerta está abierta o cerrada. Y por último, tiene un sensor de temperatura y humedad, con el cual podemos, tambien, vía notificación ser alertados si el pedido suceptible a temperatura que recibimos se esta enfriando o calentando.

Actuadores

SmartMailBox cuenta con un motor que funciona como cerradura de la puerta. En base a lo cotejado en la base de datos se abre o no. También tiene de actuadores (un relé y un LED RGB) el LED permitirá notificar visualmente si el código escaneado es correcto o incorrecto, si hubo algun error o alerta y si hay algun paquete dentro para retirar. El rele funciona como switch de corriente para el escaner de código, el rele lo alimenta de corriente si el sensor de proximidad detecta algo a una determinada distancia.





Componentes utilizados

- 1 x Arduino Mega 2560
- 1 x Módulo Bluetooth HC-06
- 1 x Lector de Codigo de Barras Rs-232
- 1x Sensor de temperatura y humedad DHT11
- 1 x Módulo de Sensor Ultrasónico HC-SR04
- 1 x Celda de carga HX711
- 1 x Sensor optico de barrera FC-03
- 1 x ServoMotor Sg90
- 1 x LED RGB
- 1 x Relevador de 10A
- 3 x Resistencia 220 ohms
- 1 x fuente de alimentacion de CC 5v 2A
- Cables

Diagramas en Bloque

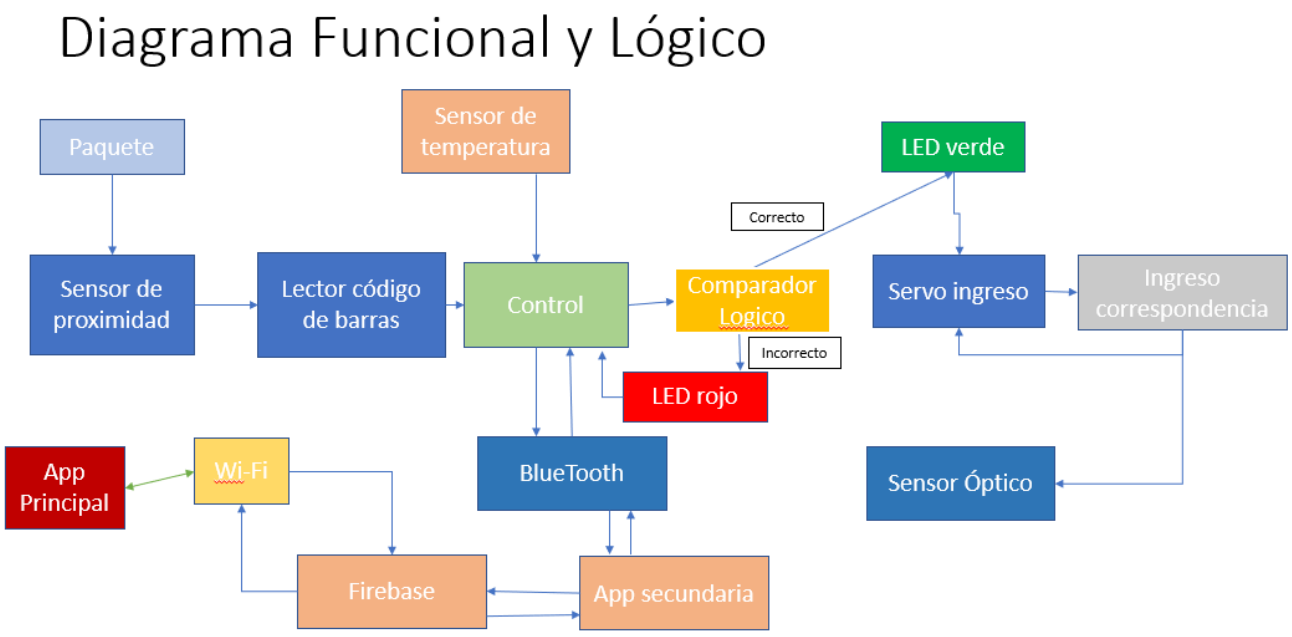
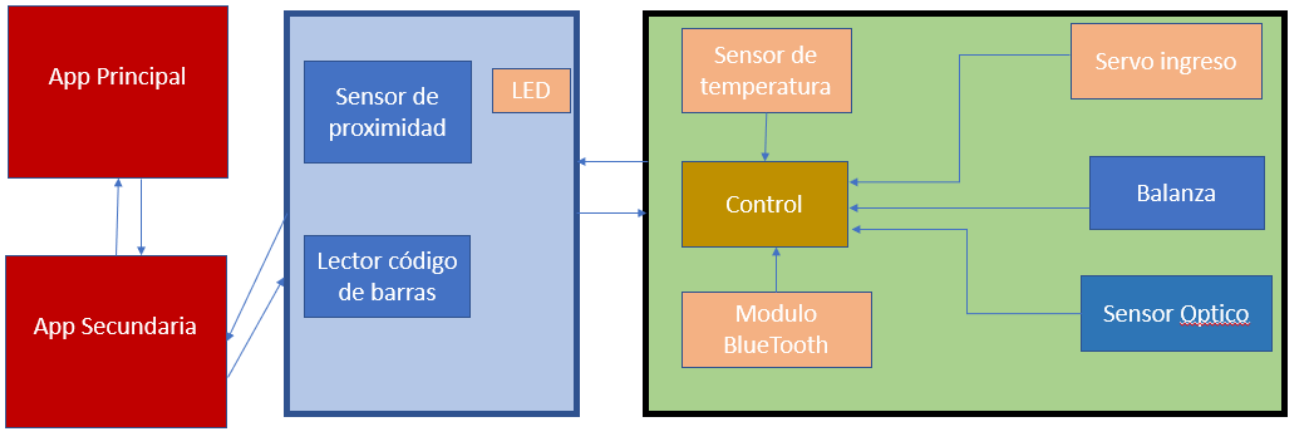


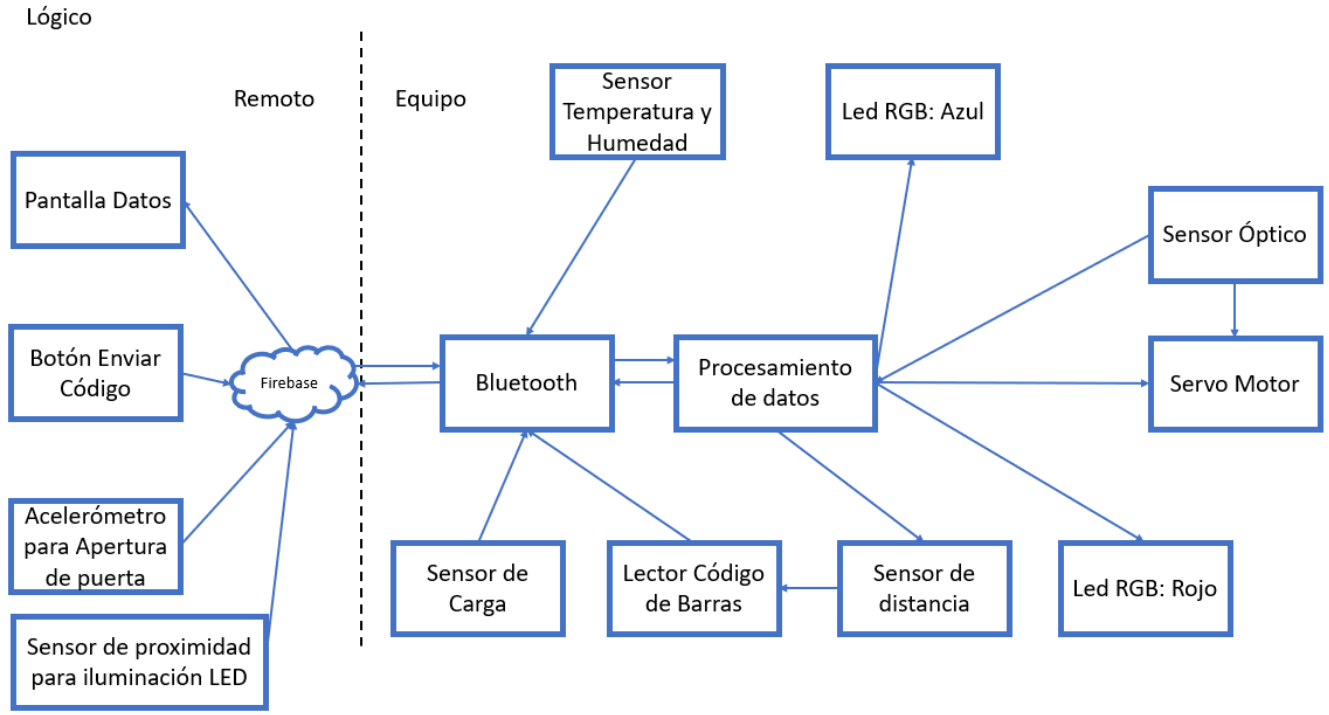
Diagrama físico

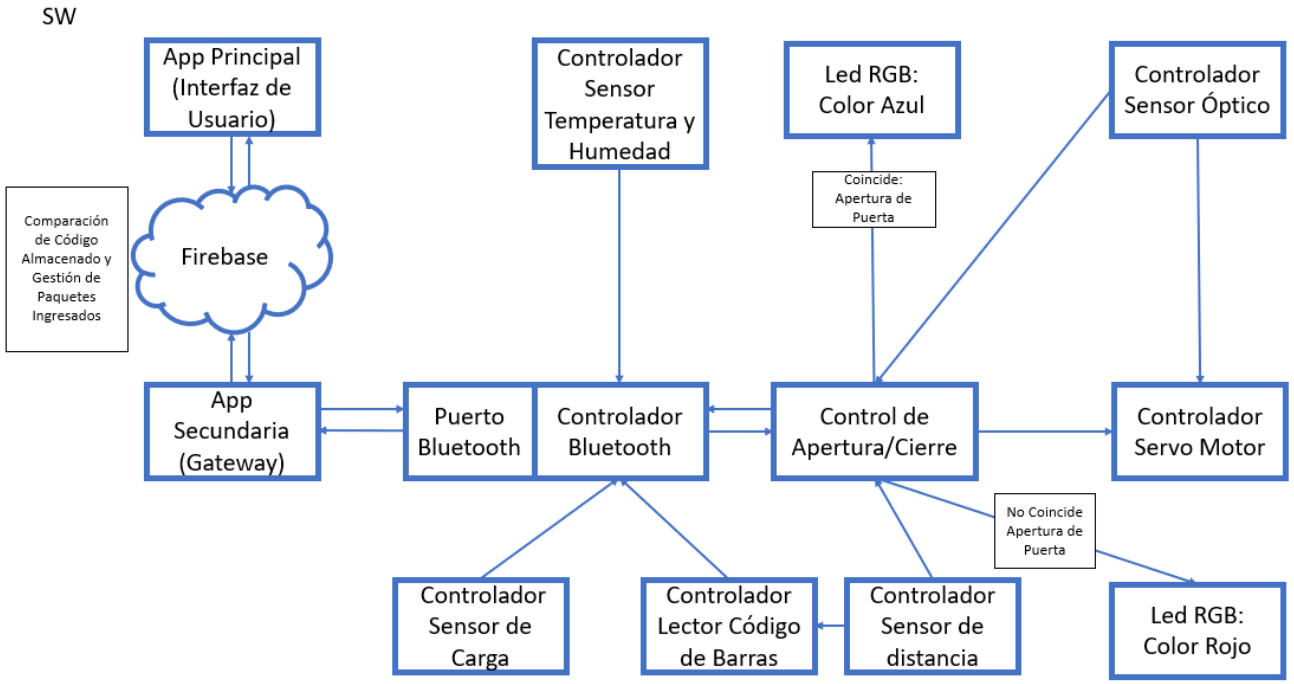


Remoto

Exterior Mailbox

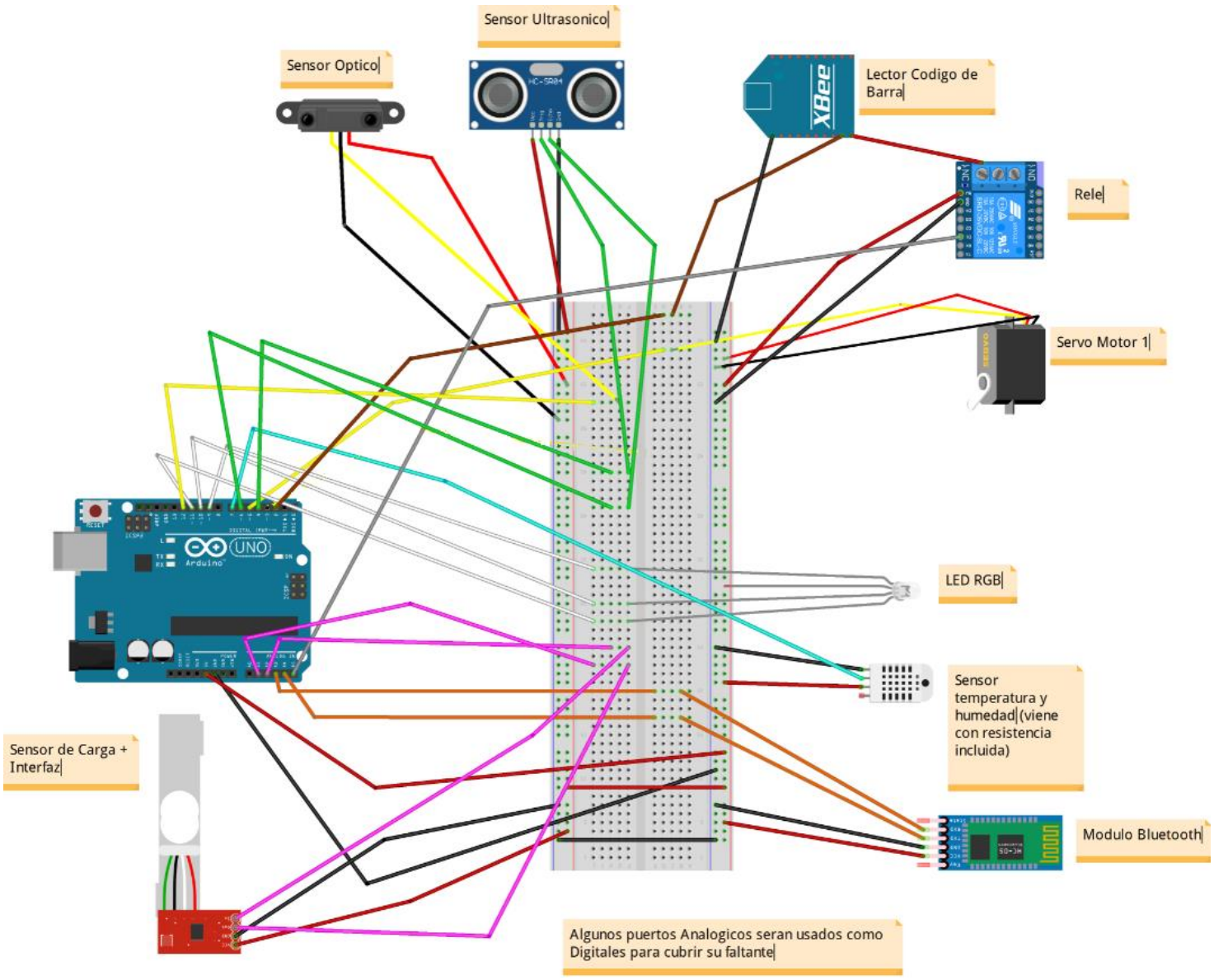
Interior Mailbox


Lógico
Remoto
Equipo
Pantalla Datos
Botón Enviar Código
Acelerómetro para Apertura de puerta
Sensor de proximidad para iluminación LED
Firebase
Sensor Temperatura y Humedad
Led RGB: Azul
Sensor Óptico
Servo Motor
Bluetooth
Procesamiento de datos
Sensor de Carga
Lector Código de Barras
Sensor de distancia
Led RGB: Rojo


SW
App Principal (Interfaz de Usuario)
Comparación de Código Almacenado y Gestión de Paquetes Ingresados
Firebase
App Secundaria (Gateway)
Puerto Bluetooth
Controlador Bluetooth
Controlador Sensor Temperatura y Humedad
Led RGB: Color Azul
Coincide: Apertura de Puerta
Controlador Sensor Óptico
Control de Apertura/Cierre
Controlador Servo Motor
No Coincide Apertura de Puerta
Controlador Sensor de Carga
Controlador Lector Código de Barras
Controlador Sensor de distancia
Led RGB: Color Rojo

Circuito

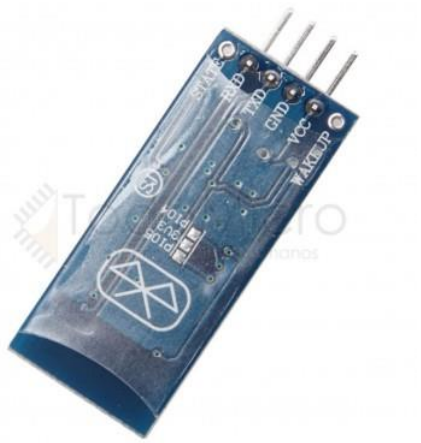
El siguiente circuito fue elaborado utilizando Fritzing (http://fritzing.org/home/)



Descripción de componentes

Modulo Bluetooth HC-06

Tanto el módulo Bluetooth utilizado para captar las señales bluetooth del ambiente como el utilizado para la comunicación con Android son módulos HC-06.
El módulo HC-06 solo puede configurarse como Slave.



Especificaciones técnicas

- Voltaje de Operación: 3.3V / 5V.
- Corriente de Operación: < 40 mA
- Corriente modo sleep: < 1mA
- Chip: BC417143
- Alcance 10 metros
- Velocidad de transmisión: 1200bps hasta 1.3Mbps
- Baudrate por defecto: 9600,8,1,n.
- Bluetooth: V2.0+EDR
- Longitud de cable: 21.5cm
- Frecuencia: Banda ISM de 2,4 GHz
- Modulación: GFSK (Gaussian Frequency Shift Keying)
- Potencia de emisión: 4 dBm, clase 2
- Sensibilidad: -84dBm a 0.1% VER
- Velocidad asíncrona: 2.1Mbps (máx.) / 160 kbps.
- Velocidad síncronos: 1Mbps/1Mbps
- Seguridad: Autenticación y encriptación
- Interfaz: Bluetooth - Puerto serie UART TTL

Conexiones

El módulo HC-06 posee dos pares importantes de pines: los de alimentación y los de transmisión.

Los pines de transmisión (**RXD** y **TXD**) se conectan a los pares de pines de un puerto serial del arduino (O en el caso de no tener un puerto serie libre, al par de pines seleccionados utilizando la librería “SoftwareSerial”). Se debe conectar el pin de transmisión del arduino con el de recepción del HC-06 y viceversa.
.

Los pines de alimentación (**VCC** y **GND**) se conectan directamente a la alimentación del arduino. El pin VCC del módulo HC-06 requiere 5V para funcionar correctamente.

Modo AT

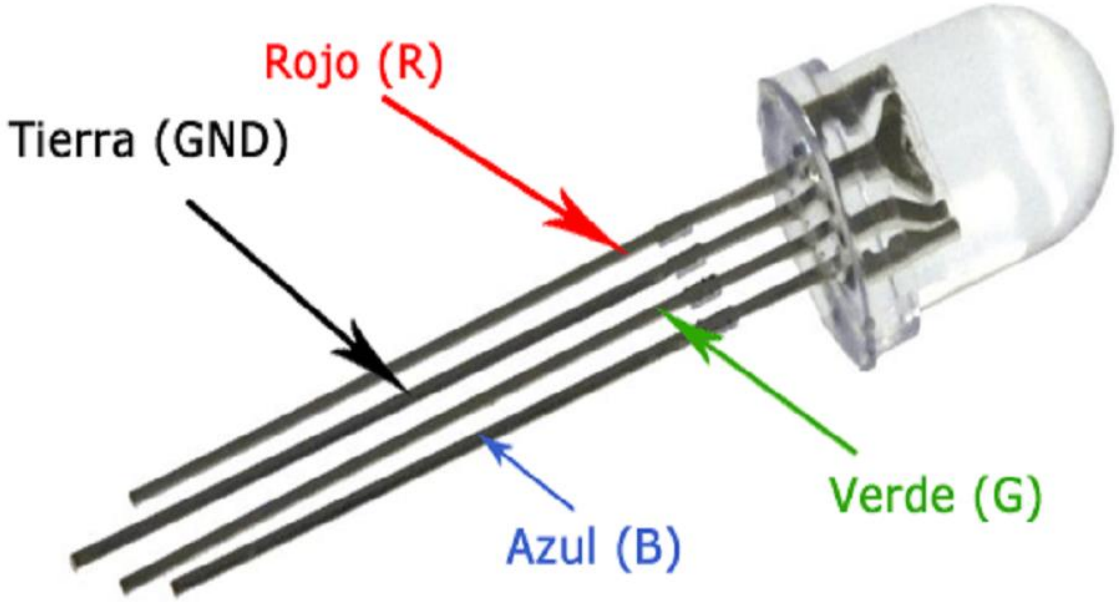
El modo AT es el estado del módulo HC-05 en el cual se pueden enviar comandos AT. Estos comandos se utilizan para configurar el módulo y para enviarle órdenes. Las diferencias visibles entre el modo normal y el AT son las siguientes: En modo AT el led del módulo parpadea mucho más lento que en modo normal, y la velocidad de transmisión en modo AT es de 38400 baudios mientras que la del modo normal es de 9600 baudios (por defecto, se puede modificar con los comandos AT)
La lista de comandos AT utilizados es la siguiente. Todos los comandos deben terminar con los caracteres “\r\n” para que el módulo HC-05 los tome correctamente.

Comando AT	**Respuesta(s) esperada(s)**	**Descripción**
AT	OK	Este comando se utiliza para saber si el HC-05 está en modo AT (Devuelve OK) o no (no devuelve nada)
AT+RESET	OK	Se utiliza para resetear el módulo HC-05
AT+ADDR	+ADDR:<dirección MAC> OK	Se utiliza para saber la dirección MAC del módulo
AT+NAME?	+NAME:<nombre> OK	Se utiliza para saber el nombre del módulo
AT+NAME=<nombre>	OK	Se utiliza para cambiar el nombre del módulo
AT+PSWD?	+PSWD:<password> OK	Se utiliza para saber la contraseña del módulo (Por defecto es “1234”)
AT+PSWD=<password>	OK	Se utiliza para cambiar la contraseña del módulo
AT+ ROLE?	+ROLE:<rol> OK	Se utiliza para saber en qué modo está el módulo. Si devuelve 0 está en modo esclavo, 1 en maestro y 2 en híbrido. Por defecto devuelve 0.
AT+ROLE=<rol>	OK	Se utiliza para configurar el modo del módulo
AT+ INQM?	+INQM=<p1>,<p2>,<p3> OK	Se utiliza para conocer los 3 parámetros del modo INQ (Que es el modo en el cual se buscan los dispositivos cercanos) p1: indica si se debe realizar el descubrimiento de

		dispositivos con modo rssi (1) o no (0). Siempre utilizamos este parámetro en 1 porque nos interesa conocer el RSSI (indicador de fuerza de la señal recibida) p2: Indica el máximo de dispositivos que se van a buscar durante el descubrimiento. p3: Indica el timeout del modo INQ. Su valor real será p3*1.28s. En caso de no encontrar lecturas acordes a los parámetros de Clases, el modo INQ se verá interrumpido por el valor real
AT+INQM=<p1>,<p2>,<p3>	OK	Se utiliza para configurar los tres parámetros del proceso de descubrimiento de dispositivos (Modo INQ)
AT+STATE?	+STATE:<estado> OK	Se utiliza para conocer el estado actual del módulo. Los estados pueden ser los siguientes: ● "INITIALIZED" > Módulo inicializado ● "READY" > Módulo listo ● "PAIRABLE" > Esperando emparejamiento ● "PAIRED" > Módulo emparejado ● "INQUIRING" > Descubriendo dispositivos ● "CONNECTING" > Conectando a un dispositivo ● "CONNECTED" > Conectado a un dispositivo ● "DISCONNECTED" > Desconectado En nuestro caso de aplicación, los estados que nos interesan son el de "módulo listo" para saber que podemos empezar a configurar el módulo y buscar dispositivos; y el de "descubriendo dispositivos" ante una espera larga para saber si es que el módulo sigue buscando dispositivos o está tildado.
AT+INIT	OK	Se utiliza para iniciar el antes mencionado "Modo INQ". No se puede realizar la búsqueda de dispositivos sin primero enviar este comando. El comando puede también devolver "ERROR(17)", que nos indica que ya habíamos enviado este comando anteriormente y ya estamos en modo INQ
AT+INQ	+INQ=<p1>,<p2>,<p3> ... OK	Se utiliza este comando para realizar la búsqueda de dispositivos. Devuelve una línea "+INQ..." por cada dispositivo encontrado y un OK al final para indicar que terminó la búsqueda. Los parámetros de cada dispositivos encontrado son los siguientes: p1: La dirección MAC del dispositivo p2: El tipo de dispositivo p3: La intensidad de la señal recibida Este comando puede también devolver "ERROR(16)", que nos indica que no hemos utilizado el comando AT+INIT aun.
AT+INQC	OK	Se utiliza este comando para cancelar la búsqueda de dispositivos

LED RGB

Entre los actuadores, utilizamos un LED para notificar, junto con la App, los cambios de estado del buzón.
Un LED es un diodo de unión PN, es decir, tiene dos electrodos, uno de ellos contiene electrones libres, y el otro huecos. Cuando se le aplica corriente, los electrones libres se re-acomodan en los huecos, liberando a su paso fotones que provocan el efecto conocido como electroluminiscencia. El colorel color de la luz generada (que depende de la energía de los fotones emitidos) viene determinado por la anchura de la banda prohibida del

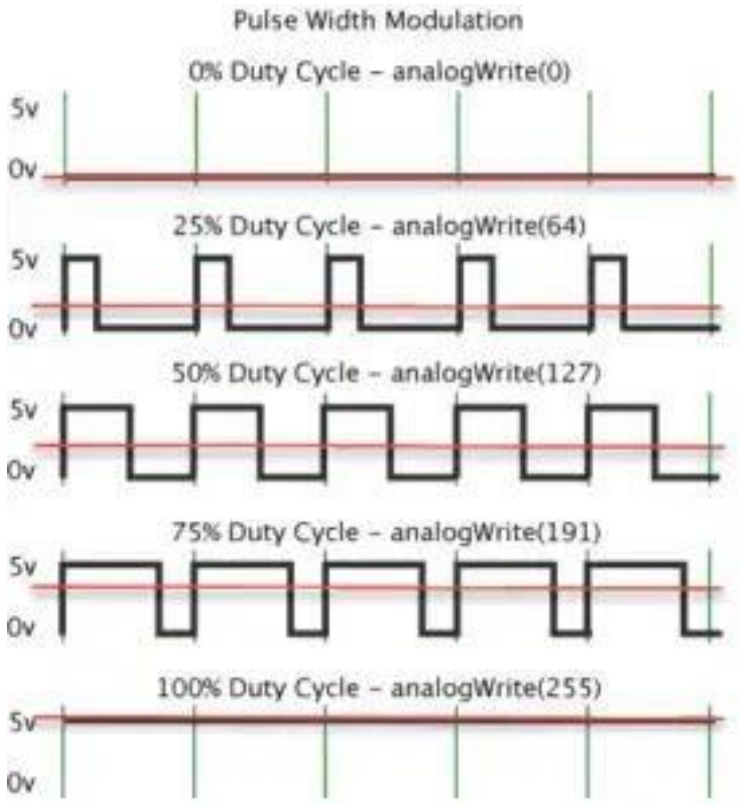


semiconductor.
Un LED RGB es en realidad la unión de tres LEDs de los colores básicos, en un encapsulado común, compartiendo el Ground (cátodo es otro nombre más para el negativo).
En función de la tensión que pongamos en cada pin podemos conseguir la mezcla de color que deseemos con relativa sencillez.

El LED que utilizamos es un LED rojo difuso, hecho de fosfuro de galio y arsénico (GaAsP), que tiene las siguiente propiedades:

- Voltajes utilizados
  Red: 2v (Aprox)
  Blue: 3.2(Aprox)
  Green: 3.2(Aprox)

Para que el LED cambie de color, tuvimos que controlar la intensidad de la luz y que color queriamos prender mediante PWM. Esto significa que para obtener el 50% de la intensidad total, tuvimos que enviar pulsos digitales, con un duty cycle del 50% del ancho del ciclo. Para una menor intensidad, el duty cycle debe ser un porcentaje menor.



En Arduino, se puede enviar dicha señal usando la función analogWrite(ledPin, value); donde value es un valor entre **0** y **255**, correspondiente al porcentaje de duty cycle que se quiere obtener en determinado PIN. En nuestro caso, como tenemos un rgb conectado cada color a un PIN diferente modificamos el valor que tomaba value para que se encienda el color deseado y la intensidad deseada.

Servomotor

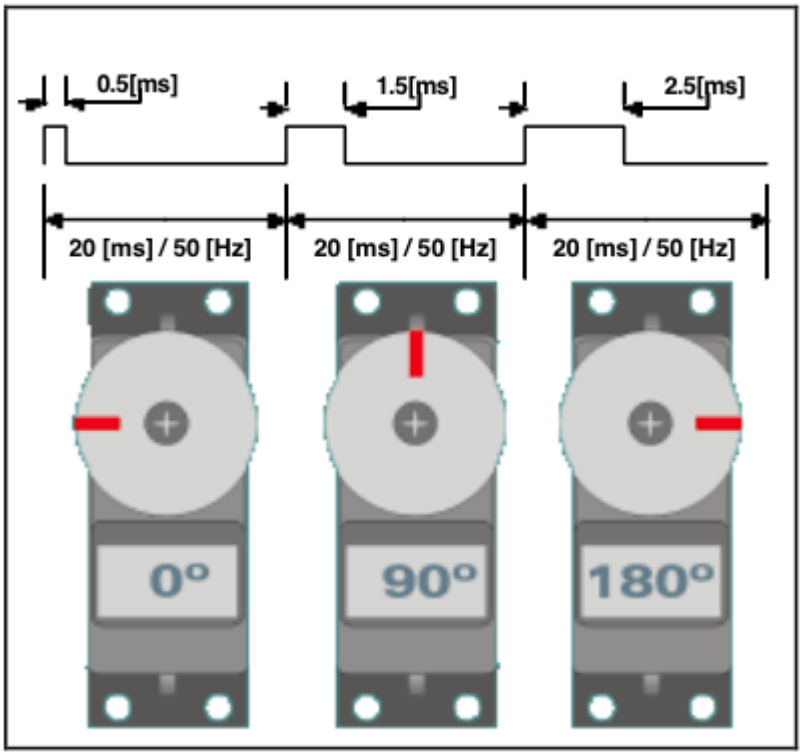
El motor servo que elegimos utilizar es el Sg90, en nuestro caso es el que cumple la funcion de cerradura, en caso de el código escaneado ser correcto el servo hace un giro de 90 grados en funcion a su posicion base (puerta trabada), lo cual permite que la puerta se destrabe y pueda ser abierta manualmente

Especificaciones técnicas

- Dimensiones (L x W xH) = 22.0 x 11.5 x 27 mm (0.86 x 0.45 x 1.0 pulgadas
- Peso: 9 gramos
- Peso con cable y conector: 10.6 gramos
- Torque a 4.8 volts: 16.7 oz/in o 1.2 kg/cm
- Voltaje de operación: 4.0 a 7.2 volts
- Velocidad de giro a 4.8 volts: 0.12 seg / 60 º
- Conector universal para la mayoría de los receptores de radio control
- Compatible con tarjetas como Arduino y microcontroladores que funcionan a 5 volts.

Modo de uso

el angulo que se le quiera dar al eje del motor servo va a depender del ancho de pulso que le enviemos por la señal digital del PIN en el cual conectamos el servo al Arduino.



Sensor optico

este sensor cumple una funcion muy importante, es la de “avisar” al arduino cuando la puerta esta cerrada o abierta, para poder empezar a realizar los procesos correspondientes de pesaje y medicion. Tambien cumple una funcion de seguridad ya que podemos saber si la puerta fue abierta sin autorizacion.

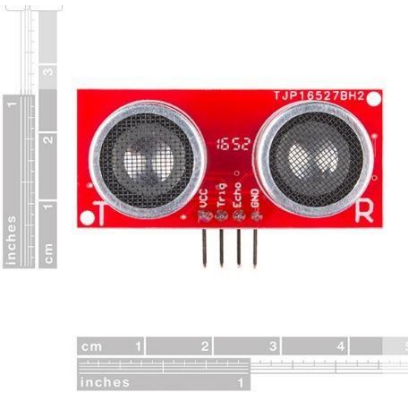
especificaciones técnicas

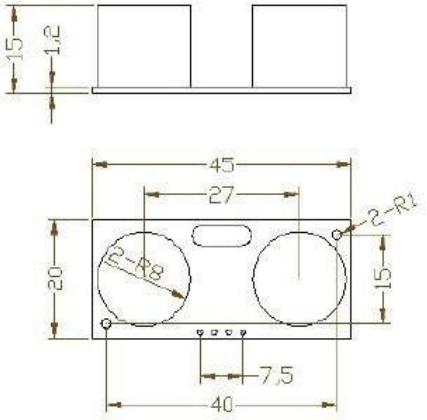
- Voltaje de Operación: 3.3V - 5V DC
- Salidas: Analogica y Digital TTL
- Sensor: MOCH22A
- Modelo Placa: FC-03 / FZ0888
- Tipo de emisor: Fotodiodo IR
- Tipo de detector: fototransistor
- Longitud de onda del emisor: 950 nm (infrarrojo)
- Peso: 8 gramos
- Dimensiones: 3.2*1.4*0.7 cm
- Ranura de 5mm
- Comparador Opamp: LM393
- Led indicador de alimentación

- Led indicador de pulso
- Salida TTL ON: Sensor bloqueado
- Salida TTL OFF: Sensor sin bloquear

Sensor Ultrasónico

Se optó por emplear un sensor ultrasónico para poder detectar los objetos frente al auto, que pudieran ocasionar colisiones.
En este caso, el modelo utilizado es el HC-SR04, siendo este uno de los más conocidos por su performance estable, y una precisión de alto rango



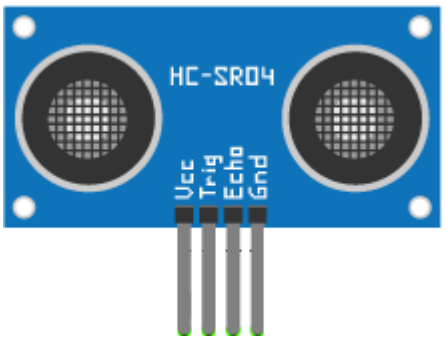


Especificaciones técnicas

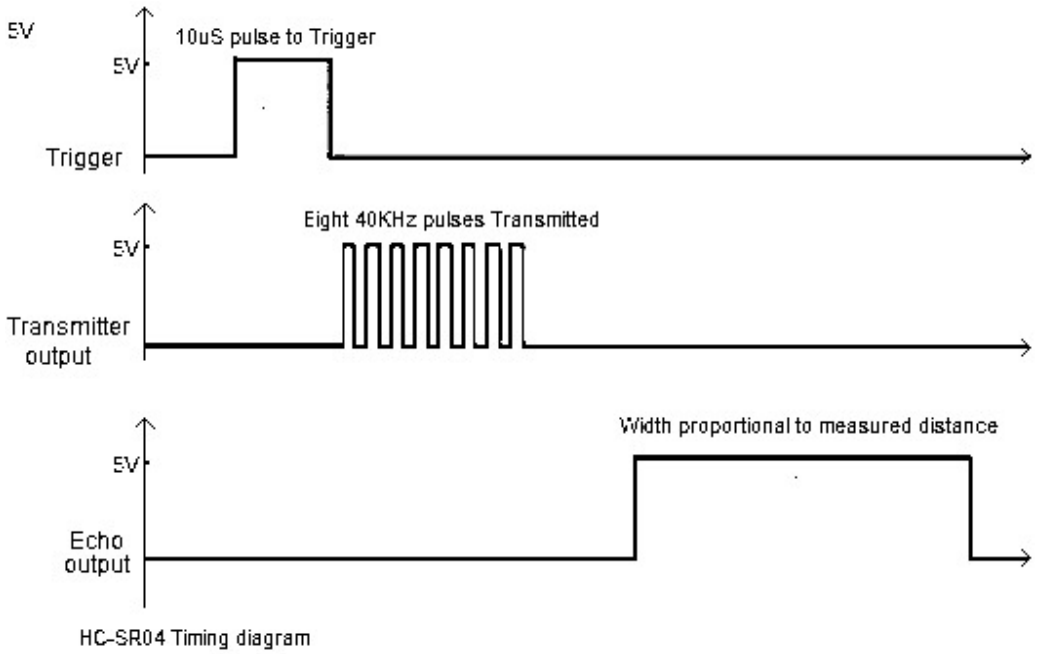
- Sensor ultrasonico para arduino y otros microcontroladores
- Tensión de operación: +5v
- Corriente nominal: 15mA
- Rango: De 2cm a 400cm (Con un error de 3mm)
- Ángulo de medición: 15 grados
- Compatible con TTL
- Dimensiones: 45*20*15mm

Conexiones

El módulo HC-SR04 dispone de dos pares de pines de conexión. Un par de pines corresponde a la alimentación del módulo (**VCC** y **GND**). El otro par de pines (**TRIG** y **ECHO**) corresponde a la medición de la distancia. Estos dos pines se conectan directamente a dos pines digitales del Arduino, el que se conecte a **TRIG** será de salida y el que se conecte a **ECHO** será de entrada.



Modo de uso



HC-SR04 Timing diagram

En lo que refiere a su funcionamiento, partiendo de un tiempo cero, se transmite un pulso ultrasónico corto, el cual en caso de impactar contra un objeto, hará que se refleje. El sensor recibe una señal y la convierte en una señal eléctrica.
Al periodo de tiempo se lo conoce como “Periodo de Ciclo”. El Periodo de Ciclo recomendado no debe ser menor a 50 milisegundos. Para operar se envía un pulso disparador (Trigger Pulse) de 10µs de ancho al pin de señal (Signal Pin). Esto hace que el modulo ultrasonico emita 8 pulsos de 40 KHz, y luego se detecta el eco que se produce en

respuesta (si no se detecta un obstáculo, el pin de salida enviará una señal alta de 38 milisegundos)
Luego se mide este ancho de pulso y se calcula la distancia siguiendo la fórmula:

$$d\ [cm] = t\ [\mu s]\ /\ v_s[\frac{\mu s}{cm}]\ /\ 2$$

Donde las variables son:

- **d** = distancia al objeto, en centímetros
- **t** = ancho de pulso del eco, en microsegundos
- $\mathbf{v_s}$ = velocidad del sonido, en microsegundos por centímetros

Esta fórmula sale de la definición de distancia, que es velocidad por tiempo. Al querer calcular la distancia a la que estamos del objeto, tenemos que recordar que el tiempo medido es el tiempo de ida y vuelta del pulso, por lo que se divide a la mitad el valor final. También invertimos las unidades de la velocidad para simplificar la fórmula dentro del código y no realizar cambio de unidades. El valor de la velocidad del sonido es 343 m/s

$$v_s = 343\ \frac{m}{s} = 34300\ \frac{cm}{s} = 0.0343\ \frac{cm}{\mu s}\ \simeq 29\ \frac{\mu s}{cm}$$

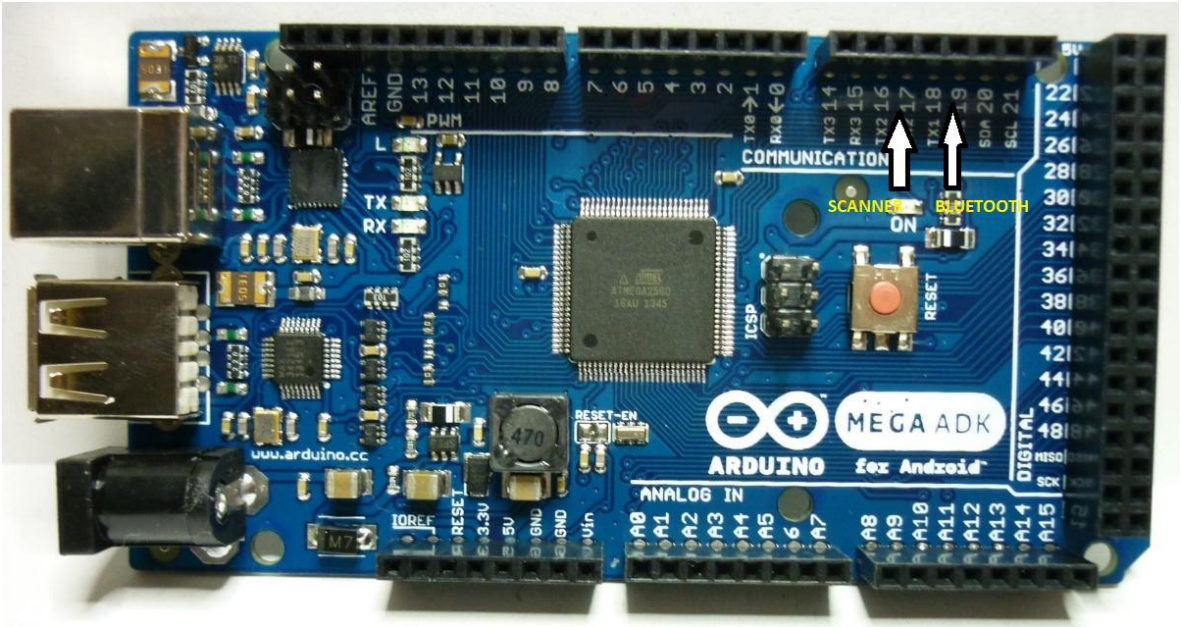
Por lo que obtenemos que la velocidad del sonido equivale a 29 microsegundos por centímetro aproximadamente, que es el valor que utilizamos en nuestra fórmula.

Elegimos estas unidades ya que para medir el ancho de pulso utilizamos la función de arduino `pulseIn`, que nos devuelve el ancho de un pulso de entrada de un pin de lectura en microsegundos. Al tener el ancho de pulso en microsegundos, decidimos modificar las unidades de la fórmula para que los valores de las constantes no sean ni muy grandes ni muy pequeños, y podemos guardar estas constantes en un entero

Problemas SE y sus soluciones

Bluetooth y scanner de códigos de barra

El problema más grande que tuvimos fue para lograr la comunicación simultanea por dos puertos serie de la placa arduino UNO que poseíamos inicialmente, solamente podíamos lograr que el modulo bluetooth haga o de receptor o de emisor si lo queriamos hacer funcionar conjuntamente y al mismo tiempo sin uso de delay o while con la lectora de barras, ya que parecieran estar usando el mismo buffer de almacenamiento del puerto serie, entonces al ser la lectora solo emisora y el bluetooth emisor y receptor , la lectora le quitaba una de las funciones al bluetooth.
Esto se soluciono cuando conseguimos un arduino MEGA 2560 que posee multiples puertos serie, asi logramos conectar la lectora de código de barras al serie 2 y el modulo blueetooth al serie 1 sin necesidad de utilizar softwareserial para indicar algun pin digital como puerto serie y no generar mas colisiones



Alimentación

Uno de los problemas más grandes que nos encontramos fue la alimentación de los circuitos. Originalmente probamos alimentar todo el USB de la computadora pero como es de menos de 1 Ampere. Con el consumo de los motores, la lectora de códigos y la placa Arduino en sí, no llegaban a encender el laser del scanner siquiera.
Una solución, que está lejos de ser la óptima, fue comprar una fuente de alimentacion externa con puerto jack de 5 voltios y 2 ampere de corriente, la cual no enchufamos directamente a la placa arduino, sino que le cortamos los cables y alimentamos las conexiones directamente del positivo.
De esta manera, si bien el consumo es muy alto y poco eficiente, logramos su funcionamiento correcto.

Por otro lado, la lectora sigue teniendo perdidas de energia durante escaneos de manera aleatoria, por lo que una solucion para el modelo final podria ser una fuente de alimentacion de 7 a 12 voltions y 2 o 3 ampere enchufada directamente en la placa arduino.

Android

Aplicación

Disclaimer: Las aplicaciones de SmartMailBox están desarrolladas para la plataforma Android, orientadas a dispositivos con Android 6.0 en adelante, cubriendo así al 62.6% de dispositivos aproximadamente.

La plataforma ofrecida por SmartMailBox está compuesta por dos aplicaciones: una aplicación para los clientes y una aplicación para el servidor. Están diseñadas para manejar los pedidos y/o compras que realice el propietario del buzón, y comunicarse con SmartMailBox para cumplir dicho objetivo.

A continuación, se detalla la funcionalidad de cada una y cómo interactúan entre sí.

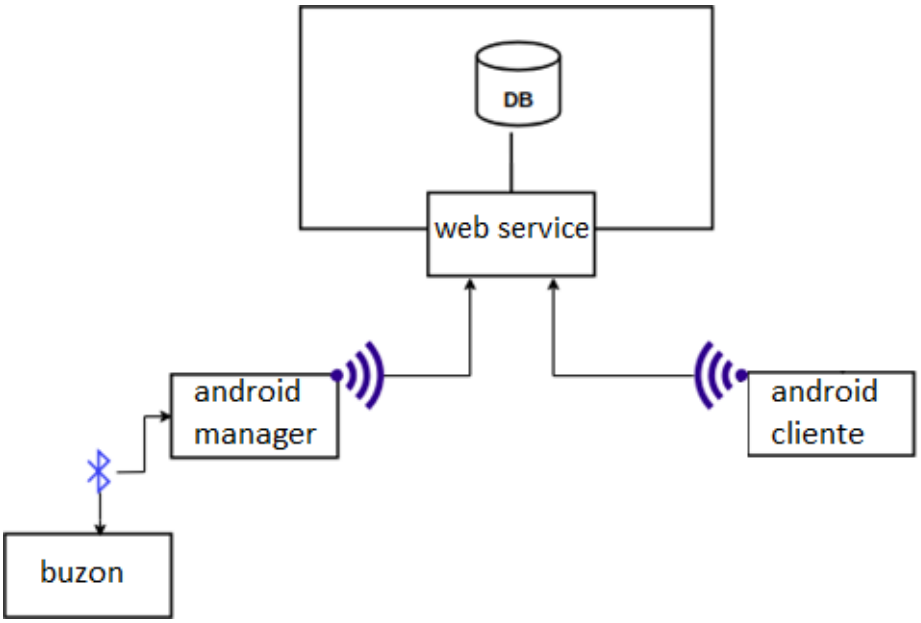
Cliente

Orientada al dueño del buzón, permite que dicho usuario pueda controlar el buzón.
Posee una primera pantalla con diferentes botones para gestionar los paquetes tanto esperados, como los recibidos y su estado.
Uno de los botones nos lleva a la pantalla general donde se visualiza el estado general del buzón. Esto es: Cantidad de paquetes dentro, humedad y temperatura, estado de la puerta, el peso total de los paquetes dentro.
Otro de los botones me permite cargar en la base de datos el código del paquete a esperar y su respectiva descripción. Además, la aplicación permite visualizar los paquetes esperados previamente cuyos códigos y descripciones fueron previamente cargados y los paquetes ya almacenados en el buzón mostrando también, el peso, individual de cada uno.

Manager

Esta aplicación cumple la función de Gateway en el buzón para otorgarle salida a la nube mediante conexión bluetooth, y así, poder realizar las comparaciones de código escaneado en el buzón con los códigos almacenados por el usuario en la base de datos. También se encarga de realizar las notificaciones correspondientes al usuario en caso de detectar alguna anomalía en el buzón.
Esta aplicación esta en constante comunicación tanto con el Arduino como con el web service, actualizando las tablas de la base de datos.

Diagrama de comunicación general



Mejoras para el producto

Tecnología Bluetooth para conexión con la nube

Si bien, actualmente SmartMailBox alcanza su destino, que sería el web service, a través del módulo Bluetooth, es decir, necesitamos de un teléfono celular con Android para poder conectar vía bluetooth al Arduino, esto se podría mejorar de la siguiente manera:

1. En lugar de tener conectado al Arduino un módulo BlueTooth se podría utilizar un modulo Wi-Fi para poder tener conexión a internet directamente sin necesidad de un teléfono móvil como intermediario.
2. Al usar modulo Wi-Fi y tener salida directa a internet ya no se necesitaría una aplicación Manager que utilizar ni emparejar vía bluetooth al arduino.
3. También se podría cambiar el web service por una base de datos online como FireBase.

Tipos de Códigos a escanear

Actualmente el prototipo de SmartMailBox escanea códigos de barra, una mejora necesaria es la de utilizar un Scanner que pueda leer distintos tipos de códigos (Barras, QR, etc.) Esto mejoraría la diversidad de paquetes que podrían recibirse, extendiéndose por ejemplo a facturas de gas, luz, teléfono y no solo limitándose a los de paquetería por internet.

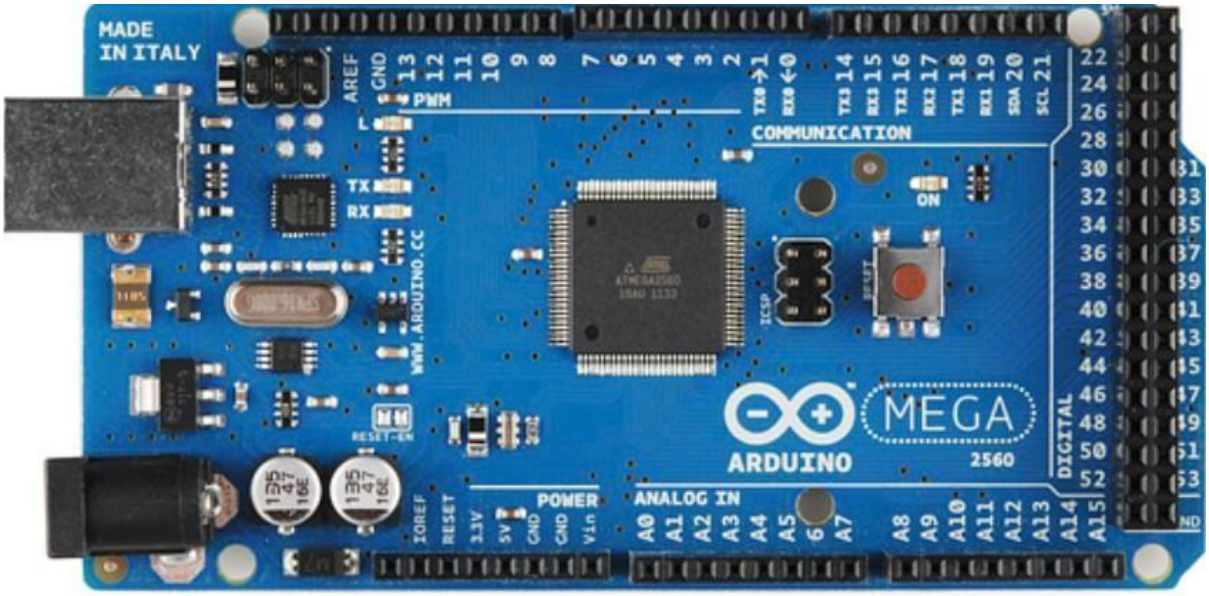
Cantidad de peso soportado.

Si bien actualmente, la celda de carga soporta hasta 10kg de peso en paquetes, una mejora podría ser colocarle una celda de carga con mayor capacidad, en caso de tener un buzón más amplio o esperar paquetes de un alto peso.

Puerta de egreso y servos.

El buzón consta de una sola puerta para ingreso y egresos de los paquetes, una mejora importante podría ser la de poner una puerta aparte, solo de uso por el dueño para quitar los paquetes y que la puerta de ingreso sea más pequeña por motivos de seguridad y que no se puedan quitar paquetes por ahí, además los servos podrían mejorarse por algunos más robustos, mejorando aún más la seguridad de las puertas, haciendo que sea más difícil forzarlas.

Anexo I: Arduino Mega 2560



Arduino Mega 2560 es una versión ampliada de la tarjeta original de Arduino y está basada en el microcontrolador Atmega2560.
Dispone de 54 entradas/salidas digitales, 14 de las cuales se pueden utilizar como salidas PWM (modulación de anchura de pulso). Además dispone de 16 entradas analogicas, 4 UARTs (puertos serie), un oscilador de 16MHz, una conexión USB, un conector de alimentación, un conector ICSP y un pulsador para el reset.

Alimentación

El Arduino Mega puede ser alimentado vía la conexión USB o con una fuente de alimentación externa. El origen de la alimentación se selecciona automáticamente.
Las fuentes de alimentación externas (no-USB) pueden ser tanto un transformador o una batería. El transformador se puede conectar usando un conector macho de 2.1mm con centro positivo en el conector hembra de la placa. Los cables de la batería puede conectarse a los pines Gnd y Vin en los conectores de alimentación (POWER)
La placa puede trabajar con una alimentación externa de entre 6 a 20 voltios. Si el voltaje suministrado es inferior a 7V, el pin de 5V puede proporcionar menos de 5 Voltios y la placa puede volverse inestable; si se usan mas de 12V los reguladores de voltaje se pueden sobrecalentar y dañar la placa. El rango recomendado es de 7 a 12 voltios.
Los pines de alimentación son los siguientes:

- **VIN:** La entrada de voltaje a la placa Arduino cuando se está usando una fuente externa de alimentación (en opuesto a los 5 voltios de la conexión USB). Se puede proporcionar voltaje a través de este pin, o, si se está alimentando a través de la conexión de 2.1mm , acceder a ella a través de este pin.

- **5V:** La fuente de voltaje estabilizado usado para alimentar el microcontrolador y otros componentes de la placa. Esta puede provenir de VIN a través de un regulador integrado en la placa, o proporcionada directamente por el USB u otra fuente estabilizada de 5V.
- **3.3V:** Una fuente de voltaje de 3.3 voltios generada por un regulador integrado en la placa. La corriente máxima soportada 50mA.
- **GND:** Pines de toma de tierra.

Memoria

El ATmega2560 tiene 256KB de memoria flash para almacenar código, de los cuales 8KB son usados para el arranque del sistema (bootloader). El ATmega2560 tiene 8 KB de memoria SRAM y 4KB de EEPROM, a la cual se puede acceder para leer o escribir con la librería EEPROM.

Entradas Y Salidas

Entradas y salidas digitales

Cada uno de los 54 pines digitales en el Mega pueden utilizarse como entradas o como salidas usando las funciones pinMode(), digitalWrite(), y digitalRead(). Las E/S operan a 5 voltios. Cada pin puede proporcionar o recibir una intensidad máxima de 40 mA y tiene una resistencia interna de pull-up (desconectada por defecto) de 20-50k Ohms. Además, algunos pines tienen funciones especializadas:

- **Serie 0 (RX) y 1 (TX),Serie 1: 19 (RX) y 18 (TX); Serie 2: 17 (RX) y 16 (TX); Serie 3: 15 (RX) y 14 (TX)**
  Usados para recibir (RX) transmitir (TX) datos a través de puerto serie TTL. Los pines Serie: 0 (RX) y 1 (TX) están conectados a los pines correspondientes del chip FTDI USB-to-TTL.
- **Interrupciones Externas: 2 (interrupción 0), 3 (interrupción 1), 18 (interrupción 5), 19 (interrupción 4), 20 (interrupción 3), y 21 (interrupción 2).**
  Estos pines se pueden configurar para lanzar una interrupción en un valor LOW(0V), en flancos de subida o bajada (cambio de LOW a HIGH(5V) o viceversa), o en cambios de valor. Ver la función attachInterrupt() para más detalles.
- **PWM: de 0 a 13.**
  Proporciona una salida PWM (Pulse Wave Modulation, modulación de onda por pulsos) de 8 bits de resolución (valores de 0 a 255) a traves de la función analogWrite().
- **SPI: 50 (SS), 51 (MOSI), 52 (MISO), 53 (SCK).**
  Estos pines proporcionan comunicación SPI, usando la librería SPI.
- **LED: 13.**
  Hay un LED integrado en la placa conectado al pin digital 13, cuando este pin tiene un valor HIGH(5V) el LED se enciende y cuando este tiene un valor LOW(0V) este se apaga.

Entradas y salidas analogicas

El Mega tiene 16 entradas analógicas, y cada una de ellas proporciona una resolución de 10bits (1024 valores). Por defecto se mide desde 0V a 5V, aunque es posible cambiar la cota superior de este rango usando el pin AREF y la función analogReference().

- **I2C: 20 (SDA) y 21 (SCL).**
  Soporte para el protocolo de comunicaciones I2C (TWI) usando la librería Wire.
- **AREF.**
  Voltaje de referencia para la entradas analogicas. Usado por analogReference().
- **Reset.**
  Suministrar un valor LOW (0V) para reiniciar el microcontrolador. Típicamente usado para añadir un botón de reset a los shields que no dejan acceso a este botón en la placa.

Comunicaciones

EL Arduino Mega facilita en varios aspectos la comunicación con la PC, otro Arduino u otros microcontroladores. El ATmega2560 proporciona cuatro puertos de comunicación vía serie UART TTL (5V). Un ATmega16U2 integrado en la placa canaliza esta comunicación serie a través del puerto USB y los drivers (incluidos en el software de Arduino) proporcionan un puerto serie virtual en el ordenador. El software incluye un monitor de puerto serie que permite enviar y recibir información textual de la placa Arduino. Los LEDS RX y TX de la placa parpadean cuando se detecta comunicación transmitida través de la conexión USB (no parpadean si se usa la comunicación serie a través de los pines 0 y 1).
La librería SoftwareSerial permite comunicación serie por cualquier par de pines digitales del Mega.
El ATmega2560 también soporta la comunicación I2C (TWI) y SPI. El software de Arduino incluye una librería Wire para simplificar el uso el bus I2C.

Programación

El Arduino Mega se puede programar con el software gratuito de Arduino.
El ATmega2560 en el Arduino Mega viene precargado con un gestor de arranque (bootloader) que permite cargar nuevo código sin necesidad de un programador por hardware externo. Se comunica utilizando el protocolo STK500 original.
También puede evitarse el gestor de arranque y programar directamente el microcontrolador a través del puerto ICSP (In Circuit Serial Programming).