|  |  |
| --- | --- |
| Resultado de imagen para imagen FPGA  access door | compiladores y librerias    Gomez Garcia, Julio Cesar |

**Introducción**

Para realizar este proyecto utilizamos una raspberry pi3, es un computador de placa reducida, computador de placa única o computador de placa simple (SBC) de bajo coste desarrollado en Reino Unido por la Fundación Raspberry Pi, con el objetivo de estimular la enseñanza de ciencias de la computación en las escuelas.

Aunque no se indica expresamente si es hardware libre o con derechos de marca, en su sección de preguntas y respuestas frecuentes (FAQs) explican que disponen de contratos de distribución y venta con dos empresas, pero al mismo tiempo cualquiera puede convertirse en revendedor o redistribuidor de las tarjetas RaspBerry Pi: por lo que se entiende que es un producto con propiedad registrada pero de uso libre. De esa forma mantienen el control de la plataforma pero permitiendo su uso libre tanto a nivel educativo como particular. Tampoco deja claro si es posible utilizarlo a nivel empresarial u obtener beneficios con su uso, asunto que se debe consultar con la fundación.



En cambio el software sí es open source, siendo su sistema operativo oficial una versión adaptada de Debían, denominada Raspbian, aunque permite otros sistemas operativos, incluido una versión de Windows 10. El diseño incluye un System-on-a-chipBroadcom BCM2835, que contiene un procesador central (CPU) ARM1176JZF-S a 700 MHz (el firmware incluye unos modos “Turbo” para que el usuario pueda hacerle overclock de hasta 1 GHz sin perder la garantía), un procesador gráfico (GPU) VideoCore IV, y 512 MB de memoria RAM (aunque originalmente al ser lanzado eran 256 MB). El diseño no incluye un disco duro ni unidad de estado sólido, ya que usa una tarjeta SD para el almacenamiento permanente; tampoco incluye fuente de alimentación ni carcasa. El 29 de febrero de 2012 la fundación empezó a aceptar órdenes de compra del modelo B, y el 4 de febrero de 2013 del modelo A.

a fundación da soporte para las descargas de las distribuciones para arquitectura ARM, Raspbian (derivada de Debían), RISC OS 5, Arch Linux ARM (derivado de Arch Linux) y Pidora (derivado de Fedora); y promueve principalmente el aprendizaje del lenguaje de programación Python. Otros lenguajes también soportados son Tiny BASIC, C, Perl5 y Ruby.

Tambien utilizamos un sensor ultrasónico Como su nombre lo indica, los sensores ultrasónicos miden la distancia mediante el uso de ondas ultrasónicas. El cabezal emite una onda ultrasónica y recibe la onda reflejada que retorna desde el objeto. Los sensores ultrasónicos miden la distancia al objeto contando el tiempo entre la emisión y la recepción.

La distancia se puede calcular con la siguiente fórmula:

Distancia L = 1/2 × T × C

donde L es la distancia, T es el tiempo entre la emisión y la recepción, y C es la velocidad del sonido. (El valor se multiplica por 1/2 ya que T es el tiempo de recorrido de ida y vuelta).

Son detectores de proximidad que trabajan libres de roces mecánicos y que detectan objetos a distancias de hasta 8m. El sensor emite impulsos ultrasónicos. Estos reflejan en un objeto, el sensor recibe el eco producido y lo convierte en señales eléctricas, las cuales son elaboradas en el aparato de valoración. Estos sensores trabajan solamente en el aire, y pueden detectar objetos con diferentes formas, superficies y de diferentes materiales. Los materiales pueden ser sólidos, líquidos o polvorientos, sin embargo han de ser deflectores de sonido. Los sensores trabajan según el tiempo de transcurso del eco, es decir, se valora la distancia temporal entre el impulso de emisión y el impulso del eco.

Este sensor al no necesitar el contacto físico con el objeto ofrece la posibilidad de detectar objetos frágiles, como pintura fresca, además detecta cualquier material, independientemente del color, al mismo alcance, sin ajuste ni factor de corrección. Los sensores ultrasónicos tienen una función de aprendizaje para definir el campo de detección, con un alcance mínimo y máximo de precisión de 6 mm. El problema que presentan estos dispositivos son las zonas ciegas y el problema de las falsas alarmas. La zona ciega es la zona comprendida entre el lado sensible del detector y el alcance mínimo en el que ningún objeto puede detectarse de forma fiable.

Para tener interaccion con el usuario utilizamos un dashboar que envía notificaciones. El nombre de el dashboar es Pushetta se hace para que sea comunicaciones enviar emisión sencillas para grupos de abonados. Funciona de una manera muy simple: como editor . Se crea un grupo temático, cada usuario que se suscribe este grupo recibirá una notificación cada vez que se coloca un mensaje.   
Se puede comparar con SMS con muchas ventajas:

Es libre de cargo

 No hay necesidades de los números de teléfono u otros datos personales

Remitente consiga precisas estadísticas sobre los abonados y los mensajes entregados

 Puede ser extendido para soportar dispositivos que no sean móviles (es decir. Smart TV, navegadores web, ...)

La integración de Pushetta en su aplicación es muy sencilla. Pushetta expone su funcionalidad con un API REST se puede utilizar en una amplia gama de casos, desde la aplicación web de la empresa para Arduino boceto.

Cómo funciona

Notificación de inserción son herramientas poderosas pero hoy en día este poder es expresado, se cree Pushetta para que estén disponibles para todos.   
Campo principal de la cual se cree (aunque no limitados a) es la IO. Tome un escenario simple: un sistema de automatización de la casa a base de Arduino donde es necesaria para notificar al usuario frente a algunos eventos. Hoy en día esta situación se puede resolver con:

 E-mail: este es el enfoque estándar y el más odiado de los usuarios (buzones están llenos de spam)

 SMS: esto da muchos problemas, obtener los números de teléfono, los costos por envío

Aplicación de encargo: desarrollador puede hacer una costumbre de aplicaciones con soporte de notificación de inserción. Costos y el tiempo pueden ser prohibitivos

Aquí es donde entra en juego Pushetta ayuda. Nuestro desarrollador de Arduino registra en el sitio web Pushetta y crea un canal llamado "Casa Alarmas". Toma clave de API de tablero de instrumentos y esto es todo lo que necesita para integrar la función de Pushetta a su boceto. La integración se realiza mediante simples llamadas a la API utilizable a partir de muchos entornos y lenguajes.   
La mayor parte se hace! La descarga de uno de Aplicaciones compañero de Pushetta y suscribirse al canal creado es todo lo que necesita para empezar a recibir la notificación de inserción en el evento desencadenado por aduanas sistema de automatización.   
También es posible enviar la notificación de la página Información del canal que puede ser útil en las pruebas, sino también para aplicaciones personalizadas que requieren notificaciones manuales.

Internet de las cosas

La internet de las cosas (IoT) es esencialmente un sistema de máquinas u objetos equipados con tecnologías de recopilación de datos, de manera que esos objetos pueden comunicarse entre sí. Los datos de máquina a máquina (M2M) que se generan tienen una amplia gama de usos, pero comúnmente se ven como una forma de determinar la salud y el estado de las cosas, inanimadas o con vida.

Las aplicaciones son desplegadas para un propósito específico, y no interactúan fuera de esa red. La verdadero IoT está donde diferentes aplicaciones se despliegan por razones específicas, y los datos recogidos desde las máquinas y objetos que se están monitoreando se ponen a disposición de aplicaciones de terceros. La expectativa es que la verdadera IoT proporcionará más valor del que se puede derivar de las islas separadas de información,

Su trascendencia puede ser brutal tanto en ámbitos económicos como sociales. Mayor incluso que la era digital. Y es que el Internet of Things permite que los ordenadores interactúen con elementos de la vida real y ganen independencia de los seres humanos, dejándonos a nosotros al mando de lo realmente importante.

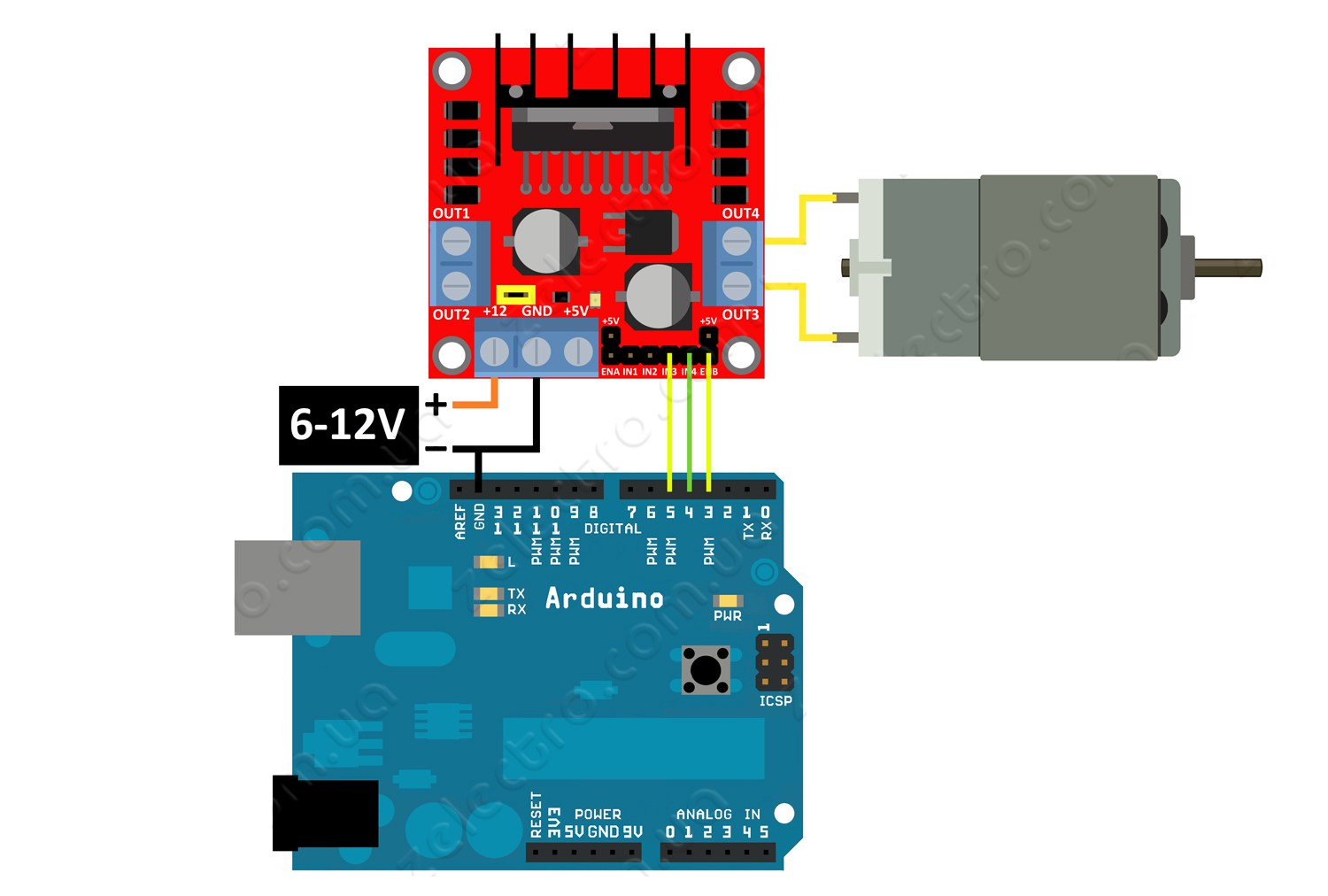
**Descripción del circuito.**

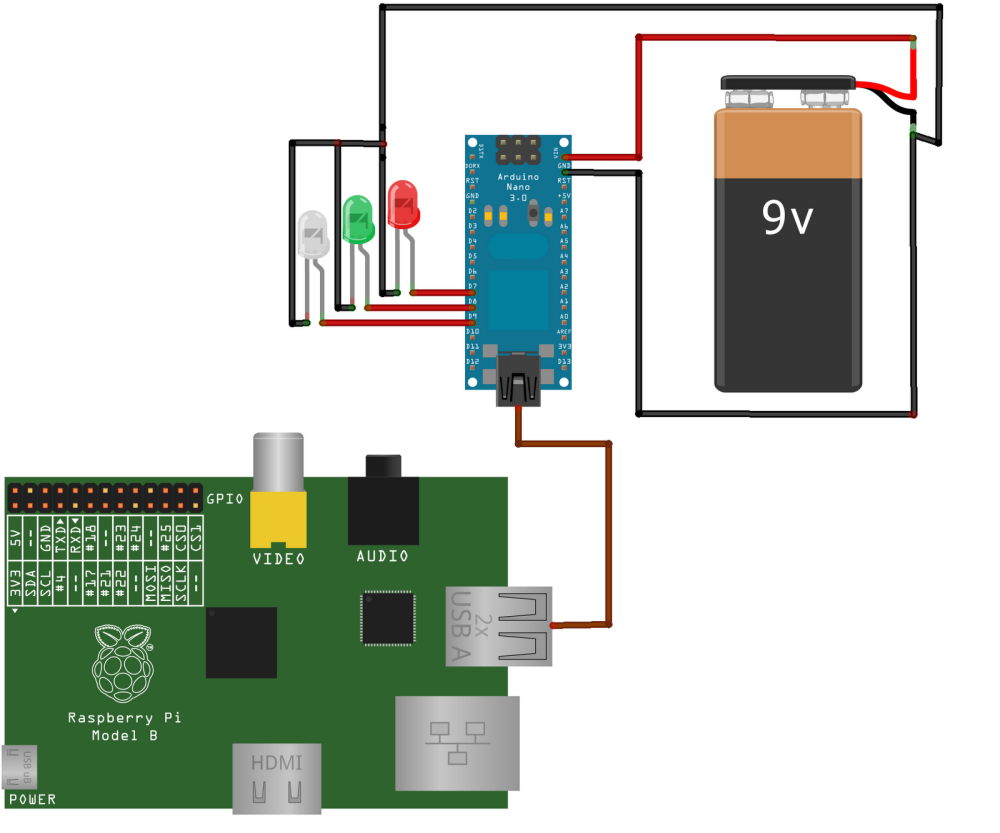
El desarrollo del proyecto nació de la necesidad de tener un control de las áreas a las que nuestras mascotas tendrán o no acceso. Esta estrechamente ligado a lo que es internet de las cosas (IoT).

Utilizamos varios dispositivos tales como una Raspberry Pi3, un Arduino Nano, puente H L298, sensor ultrasónico HC-SR04, motor de DC y un DipSwitch. Usamos la Raspberry como servidor y hace el manejo de todas las peticiones, el arduino solo ejecuta lo que le manda la Rpi, el puente H se utiliza para hacer girar un motor en ambos sentidos y simular que se está abriendo o cerrando la puerta, el sensor ultrasónico está midiendo a cada instante la distancia que hay entre el objeto y el sensor y si se cumple la condición ejecuta ciertas acciones, por último el dipswitch lo utilizamos como simulador para los diferentes sensores que no pudimos conseguir como finales de carrera, sensor de peso y sensor óptico que sería usado para verificar que no esté nada obstruyendo la puerta.

El funcionamiento general del circuito es el siguiente. En la Raspberry se confirma al inicio si la mascota tendrá acceso a el área que se determinó previamente. Una vez que se determinó la acción a realizar el arduino nano ya estará corriendo la parte del programa donde se estará verificando que la distancia sea menor a 15cm y que se cumpla el peso que se determinó (en nuestro caso un dipswitch), ya que se cumplan estas condiciones el arduino pregunta si la mascota tiene acceso de ser afirmativo se continua con la ejecución del resto de código si no sigue con las comparaciones. En caso afirmativo el motor sube la puerta hasta que se presenta la señal de final de carrera en alto. Llegando a la posición final alta espera 5 segundos a que entre la mascota para comenzar el descenso, girando el motor en sentido contrario una vez llegando al final de carrera en bajo se reinicia el código.

En dado caso que no llegue al final de carrera en bajo y se hubiera presentado una interrupción o bloqueo en la puerta, esta vuelve a subir hasta el final de carrera, espera un segundo y vuelve a descender, si esto ocurre 3 veces se manda una alerta directamente a el celular a través de la aplicación de pusheta.





**Código (Línea por línea).**

**Código Arduino**

|  |
| --- |
| /\* Declaramos las variables \*/ |
|  | long distancia; |
|  | long tiempo; |
|  | void finca(); |
|  | void fincb(); |
|  | int i=0; |
|  | char c; |
|  |  |
|  | void setup(){ |
|  | Serial.begin(9600); |
|  | pinMode(9, OUTPUT); //Declaramos el pin 9 como salida del pulso ultrasonico trig |
|  | pinMode(8, INPUT); // Declaramos el pin 8 como entrasa (tiempo que tarda en volver) echo |
|  | pinMode(12, OUTPUT); //led |
|  | pinMode(3, INPUT); //peso |
|  | pinMode(6, INPUT); //final de carrera alto |
|  | pinMode(7, INPUT); // final de carrera bajo |
|  | pinMode(4, OUTPUT); //motor1 |
|  | pinMode(5, OUTPUT); //motor2 |
|  | pinMode(10, OUTPUT); //obstrucción |
|  | } |
|  |  |
|  | void loop(){ |
|  | if (Serial.available()); |
|  | { |
|  | int peso, fca, fcb, inter; |
|  |  |
|  | i = 0; |
|  | peso = digitalRead(3); |
|  | fca = digitalRead(6); |
|  | fcb = digitalRead(7); |
|  |  |
|  | /\* Se estabiliza el sensor \*/ |
|  | digitalWrite(9,LOW); |
|  | delayMicroseconds(5); |
|  | /\* Se envia el pulso ultrasonico \*/ |
|  | digitalWrite(9, HIGH); |
|  | delayMicroseconds(10); |
|  | /\* Mide el tiempo transcurrido entre la salida y la llegada del pulso ultrasonico \*/ |
|  | tiempo=pulseIn(8, HIGH); |
|  | /\* Se calcula la distancia on esta formila\*/ |
|  | distancia= int(0.017\*tiempo); |
|  |  |
|  | if ((distancia < 15) and (peso == HIGH)) |
|  | { |
|  | //preguntar a RBP si es posible abrir |
|  | Serial.print("p"); //envia a la rbp p=pregunta |
|  | delay(10); |
|  | c = Serial.read(); |
|  | if (c == '1') |
|  | { |
|  | inter = digitalRead(10); |
|  | finca(fca); |
|  | digitalWrite (4, LOW); |
|  | delay(5000); |
|  | fincb(fcb, inter, fca); |
|  | digitalWrite (5, LOW); |
|  | } |
|  | } |
|  |  |
|  | } |
|  | } |
|  | void finca(int fca){ |
|  | while ( fca == LOW) |
|  | { |
|  | digitalWrite (4, HIGH); |
|  | digitalWrite (5, LOW); |
|  | fca = digitalRead(6); |
|  | } |
|  | } |
|  | void fincb(int fcb, int inter, int fca){ |
|  | while ( fcb == LOW) |
|  | { |
|  | digitalWrite (4, LOW); |
|  | digitalWrite (5, HIGH); |
|  | fcb = digitalRead(7); |
|  | inter = digitalRead(10); |
|  | if( (inter == HIGH)) |
|  | { |
|  | delay(500); |
|  | i ++; |
|  | if (i>2) |
|  | { |
|  | while ( fca == LOW){ |
|  | digitalWrite (4, HIGH); |
|  | digitalWrite (5, LOW); |
|  | fca = digitalRead(6); |
|  | } |
|  | digitalWrite (4, LOW); |
|  | Serial.print("9"); //mandar senial a la raspberry |
|  | break; |
|  | } |
|  | finca(fca); |
|  | } |
|  | } |
|  | } |

**Código Raspberry**

|  |
| --- |
| Importserial |
|  | import time |
|  | import push |
|  |  |
|  | arduino = serial.Serial('/dev/ttyUSB1', 9600) #ttyUSB0 para arduino nano , ttyACMo para arduino uno |
|  | arduino.open |
|  |  |
|  | txt = '' |
|  | print("........Bienvenido.......\n") |
|  | print("Configuracion:\n") |
|  |  |
|  | access = int(raw\_input("El perrito tendra acceso a la puerta de la cocina? 0=NO, 1=SI ")) #acceso |
|  |  |
|  | print("........Configuracion completa.......\n") |
|  | if access == 1: |
|  | print("El perrito tiene acceso a la cocina \n") |
|  | if access == 0: |
|  | print("El perrito NO tiene acceso a la cocina \n") |
|  |  |
|  | while True: |
|  | time.sleep(0.1) |
|  | while arduino.inWaiting() > 0: |
|  | txt = arduino.read(1) |
|  | print txt |
|  | if txt == "p": |
|  | if access == 1: |
|  | arduino.write("1") |
|  | else: |
|  | arduino.write("0") |
|  | elif txt == "9": |
|  | push.sendNotification("8969e7c6b9d30c765ab2dea18fbed44a244a880d", "access-door", "Puerta bloqueada :( ") |
|  | txt = '' |
|  | arduino.close() #Finalizamos la comunicación |

**Código Pusheta**

|  |
| --- |
|  |
| #Importamos algunas librerias necesarias |
|  | import urllib2 |
|  | import json |
|  |  |
|  | #Esta es la funcion que usa el script para enviar las notificaciones |
|  | def sendNotification(token, channel, message): |
|  | data = { |
|  | "body" : message, |
|  | "message\_type" : "text/plain" |
|  | } |
|  |  |
|  | req = urllib2.Request('http://api.pushetta.com/api/pushes/{0}/'.format(channel)) |
|  | req.add\_header('Content-Type', 'application/json') |
|  | req.add\_header('Authorization', 'Token {0}'.format(token)) |
|  |  |
|  | response = urllib2.urlopen(req, json.dumps(data)) |
|  |  |