DEPARTAMENTO DE INGENIERÍA E INVESTIGACIONES TECNOLÓGICAS

Sistemas Operativos Avanzados

Sistemas Embebidos (IoT) + Android

“Reconocimiento Facial”

Comisión: Miércoles Noche

Ciclo Lectivo: Primer Cuatrimestre - Año 2018

Docentes:

- Lic. De Luca Graciela
- Ing. Valente Waldo
- Ing. Carnuccio Esteban
- Ing. Volker Mariano
- Ing. Garcia Gerardo

Alumnos:

- Zurdo, Misael — DNI: 39.272.050
- Avalos, Leonel Martin — DNI: 39.184.666
- Callapiña, Guillermo — DNI: 39.185.941
- Crescente, Maximiliano — DNI: 35.959.751

Índice

Objetivo

Durante la cursada se planteó como objetivo la creación de un sistema embebido que funcione en conjunto a un aplicativo en Android, como se haría en un sistema de internet de las cosas.

Para esto, desarrollamos un sistema embebido (SE) que se encarga del reconocimiento facial mediante la obtención de datos del ambiente (foto, movimiento). En caso de reconocer un rostro, el sistema reaccionará mostrando el nombre de la persona, levantará una barrera y encenderá una alarma tanto sonora como lumínica.
Desde la aplicación el usuario de ***ReconocientoFacial*** podrá también seleccionar la persona que quiere entrar.
El sistema embebido estará dispuesto sobre una mesa. Siendo como posibles ubicaciones entradas y salidas a sitios, etc.

Descripción del Entorno

Hardware

Sistema Embebido

- Router TL-WR941ND
- Raspberry Pi 3 Model B
- Tarjeta micro SD 32 GB
- Protoboard
- 1 x Logitech Webcam C170
- 1 x Sensor de movimiento PIR (HC-SR501)
- 1 x LED 5V
- 1 x Display
- 1 x Buzzer Activo
- 1 x Servomotor SG90

Aplicación Android

- Notebook Acer Aspire Es15 (Intel Core i3 - 4 GB Ram - SO Windows 10 SP1)
- Motorola z play (Android 7.1.1)
- Sensor de Luz
- Acelerómetro
- Sensor de pasos

Software

Sistema Embebido

- IDE: Visual studio code
- Nodejs 8.11.0
- Librerías principales
    - Johnny five (es la plataforma que permite utilizar JavaScript para robótica e IoT)
    - Face-recognition (permite entrenar e identificar rostros)
    - Opencv4nodejs (permite utilizar opencv desde nodejs)
    - Socket.io-client (permite utilizar websocket fácilmente como cliente)
- Opencv
- Fritzing
    - Permite realizar diseños electrónicos.

Web Service

- IDE: Visual studio code
- Nodejs 8.11.0
- Librerías principales
    - Socket.io (permite utilizar websocket fácilmente como servidor)
    - Express (permite generar un web service fácilmente con nodejs)

Aplicación Android

- IDE: Visual Studio Code
- Reac Native
    - Esta tecnología permite crear app tanto para Android como para iOS utilizando JavaScript.
- Librerías principales
    - React-native-sensor-manager (permite utilizar los sensores de un dispositivo android junto con react native)
    - Socket.io-client (permite utilizar websocket fácilmente como cliente)

Detalles técnicos de los sensores utilizados

- **Sensor PIR:**
  Los PIR más frecuentes son sensores de movimiento, y para ello están divididos en dos mitades de forma que detecten el cambio de radiación IR que reciben uno y otro lado, disparando la alarma cuando perciben ese cambio.
  En el SE es el encargado de controlar el movimiento delante de él y lo informa a la aplicación Android.
    - Voltaje: 5V ~ 12 V
    - Salida: 0 a 3.3 V
    - Angulo de detección: cono de 110°
    - Ajuste de parámetros: mediante 2 potenciómetros, el usuario puede modificar tanto la sensibilidad como la distancia de detección del PIR.
        - Ajuste rango de distancia: de 3 a 7 metros
        - Ajuste de retraso de tiempo: de 3 segundos a 5 minutos.

Detalles técnicos de los actuadores utilizados

- **Servomotor SG90** - PWM:
  El servomotor espera un tren de pulsos que se corresponde con el movimiento a realizar. La electrónica dentro del servomotor responderá al ancho de la señal modulada, es decir que el movimiento de este dispositivo se realiza mediante una señal PWM. Su posición de reposo es en el que se encontraba cuando se detuvo por última vez. El requerimiento de energía es bastante bajo y se permite alimentarlo con la misma fuente de alimentación que el circuito de control.
  El período del PWM es de 20 ms (50Hz). Mientras que los ciclos de trabajo que deben de usarse para realizar el giro del servomotor son los siguientes:
  - 1 ms: -90º (movimiento con sentido horario).
  - 1.5 ms: 0º (estado neutro).
  - 2 ms: +90º (movimiento con sentido anti horario).
  - Voltaje: 4.8V a 6V.
  - Peso: 14.7 gramos.
  - Torque: 2.5 Kg/cm.
  - Velocidad: 0.1 seg/60º.
  - Giro: 180º (-90º +90º).

- **Led** – PWM
  Diodos emisores de luz utilizados como actuadores, para mostrar una respuesta visual positiva relacionada con el correcto reconocimiento facial.
  Se debe utilizar una resistencia para conectarlos ya que si la tensión es inferior a un valor determinado (Valor de polarización directa), el Led no se enciende y por el contrario si es superior a este valor el Led se rompe. El led es un diodo, por lo que tiene polaridad, entonces se debe conectar de una manera específica.
  Para iluminar los Leds se utiliza PWM (Modulación por ancho de pulso), que es una técnica en la que se modifica el Duty Cycle, de la señal para controlar la energía que se envía.
  - Corriente: 20mA
  - Ground (-): 0v
  - Voltaje: 3v – 3.6v

- **Buzzer** - Digital
  Es un transductor que se encarga de transformar energía eléctrica en acústica.
  Posee dos terminales, una positiva donde se envía la señal eléctrica y otra negativa por lo general puesta a tierra. Utilizamos un buzzer activo que incorpora un oscilador simple por lo que únicamente es necesario suministrar corriente al dispositivo para que emita sonido.
  Su composición:
  - Ground (-): 0v
  - Voltaje: +5v

- **Display** - I2C
  Es un LCD (Liquid Crystal Dysplay) de 16x2, esto quiere decir que dispone de 2 filas de 16 caracteres cada una. Este LCD dispone de un módulo I2C integrado, con este podemos comunicarnos con el display con este protocolo de manera serial.

  El I2C es un bus maestro-esclavo. La transferencia de datos es siempre inicializada por un maestro (SE); el esclavo reacciona (display).
  El I2C precisa de dos líneas de señal:
  - SCL (System Clock): es la línea de los pulsos de reloj que sincronizan el sistema.
  - SDA (System Data): es la línea por la que se mueven los datos entre los dispositivos.

  I2C utiliza comunicación del tipo half-duplex por lo que no se puede enviar y recibir al mismo tiempo, por eso usa solo un pin de data (SDA).

  Además, la pantalla nuestra cuenta con 2 pines más:
  - VCC (Voltaje): es de 5 volts y está conectado a uno de los pines de la placa.
  - GND (Masa): también está conectado directamente a uno de los pines de la placa.

Alcance del Sistema

Sistema Embebido

1. El sistema embebido debe sacar fotos de rostros del ambiente.
2. El sistema embebido debe contener un registro de rostros conocidos.
3. El sistema embebido debe reaccionar a los eventos realizados por el web service mediante la utilización de socket.
4. El sistema embebido debe mostrar el nombre del rostro de la foto, en caso de conocerlo. Si no, mostrar el mensaje de “persona desconocida”.
5. El sistema embebido debe accionar el Buzzer, el Led y el Servomotor cuando la foto se encuentre entre los rostros registrados.

Aplicación Android

1. La aplicación Android debe informar la persona que quiere entrar al web service.
2. La aplicación Android debe conectarse mediante socket al web service.
3. La aplicación Android debe permitir seleccionar la persona que quiere entrar.
4. La aplicación Android debe implementar el sensor de luz para determinar una cantidad de luz optima en el ambiente.
5. La aplicación Android debe implementar el Acelerómetro del celular para el shake y confirmar que la persona quiere entrar.
6. La aplicación Android debe implementar el sensor de pasos y al detectar una cantidad minina de pasos (5).
7. La aplicación habilita la función de abrir puerta solo si existe una cantidad de luz mínima, se detectó el shake y la cantidad mínima de pasos.

Web Service

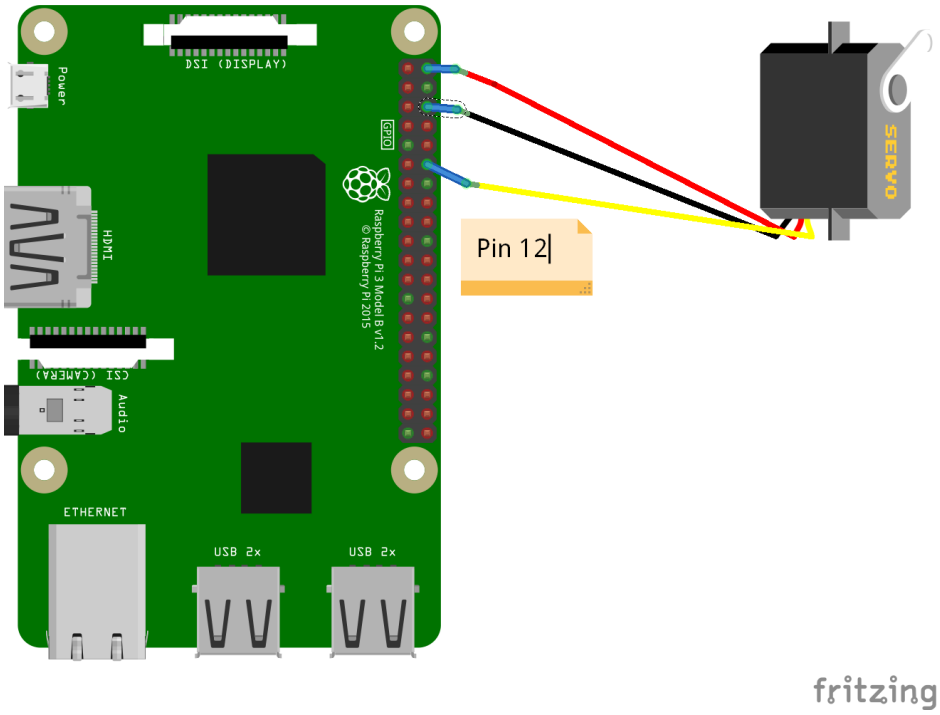
1. El servicio debe proporcionar una comunicación vía socket entre los este, el sistema embebido y la aplicación.
2. El servicio debe estar expuesto en internet.

Diseño

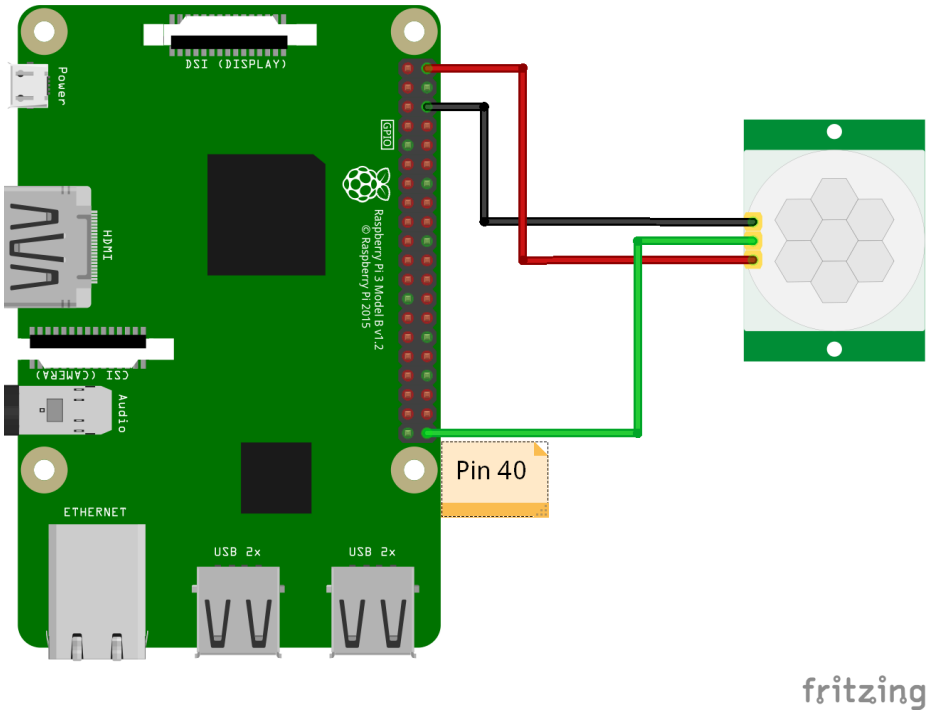
En la etapa inicial del proyecto, tuvimos algunos inconvenientes con las conexiones de sensores y actuadores, relacionados a la identificación correcta de los pines a utilizar. Después de investigar, logramos tomar conocimientos básicos de electrónica, los cuales nos empezaron a dar una idea de cómo realizar las conexiones necesarias para que nuestro Sistema Embebido vaya tomando forma.
Por lo que decidimos utilizar alguna herramienta como ***Fritzing***, la cual nos permitió tener un documento de las conexiones electrónicas de algunos sensores y actuadores.

Esquemas gráficos:

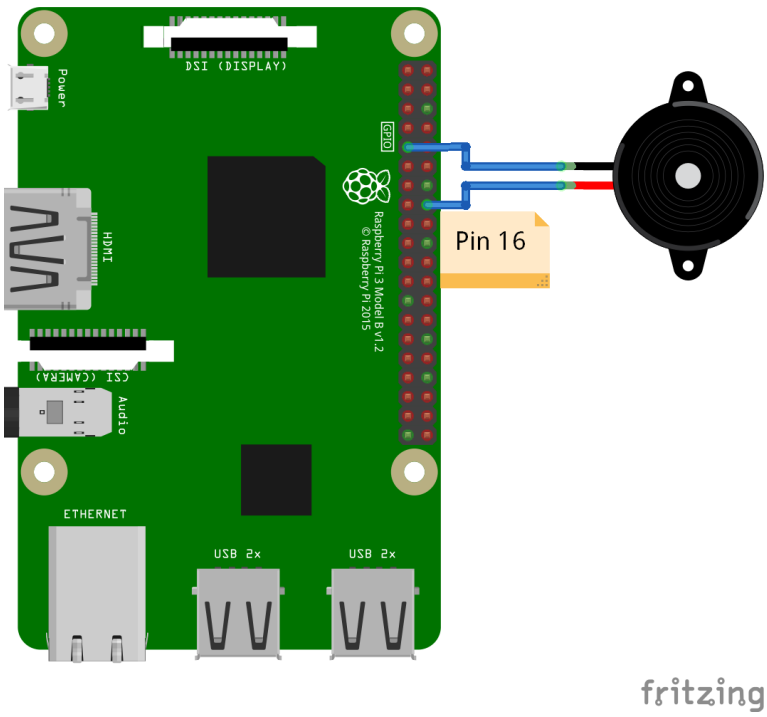
Servomotor:



Sensor de Movimiento:



Buzzer:



Implementación

Sistema Embebido

Configuración de Red

Conseguimos un router TP Link que le asignamos la dirección IP de las Raspberry. De esta manera, cada vez que conectemos la placa por señal WIFI al router, este le provee la misma dirección IP (192.168.1.102).

Instalación Sistema Operativo Raspbian

Se instaló sobre la tarjeta microSD el sistema operativo Raspbian, dedicado exclusivamente a Raspberry. El mismo está basado en Debian.

- Descargar el sistema operativo Raspbian de la página oficial
- Formatear la tarjeta micro SD. Nosotros utilizamos el programa SDFormatter V4.0
- Instalar Raspbian. Nosotros utilizamos el programa Win32DiskImager para montar la imagen a la Raspberry.

Exponer REST Service

Se utilizo nodejs para crear un web service, y utilizando una maquina virtual en la plataforma de Google de manera gratuita fue expuesta al internet dentro de una máquina virtual. Esto permite que la app y sistema embebido se conecten de cualquier lugar del mundo siempre en cuando estén conectadas al internet.

Implementar Actuador Buzzer

Simplemente instanciamos un objeto Piezo de la biblioteca johnny-five y se conectó el buzzer al pin 16. Para hacerlo sonar se ejecutará el método piezo.play() (Dentro utilizamos una pequeña melodía)

Implementar Actuador LED

Instanciamos un objeto Led de la biblioteca johnny-five y se conectó el led al pin 33 que utiliza PWM, esto nos permite utilizar el método led.pulse(); que enciende el led mediante pulsos luego para apagarlo utilizamos led.stop(); para terminar el pulso y luego led.off(); para apagar el led.

Implementar Sensor de Movimiento

Utilizamos la Instancia de un objeto Motion de la biblioteca johnny-five y se conectó el led al pin 40, e iniciamos los eventos para escuchar al sensor como motionstart y motionend que devuelve un objeto con un campo que contiene el tiempo en que se dio una respuesta y otro campo booleano que es “True” si se detectó el movimiento.

Implementar Display

En este caso utilizamos el objeto LCD de la biblioteca johnny-five y en vez de utilizar el paramos pin para marcar el pin utilizado, mandamos solo el parámetro “controller” con el valor “LCM1602” y el led debe estar conectado al pin 3(SDA) y al pin 5(SCL) además de estar conectado a la tierra y a la alimentación.

Implementar Webcam

La cámara simplemente se conecta mediante USB y al ser la única cámara conectada, mandamos de para parámetro que utilizaremos al cámara 0 cuando utilizamos opencv.

Aplicación Android

Implementar Sensor de Luz

Utilizando la librería react-native-sensor-manager tenemos acceso a un objeto llamado SensorManager que nos permite configurar el tiempo de escucha de los sensores en milisegundos, luego iniciamos un Listener de tipo “LightSensor” que nos devuelve la intensidad de luz detectada.

Implementar Acelerómetro

Utilizando la librería react-native-sensor-manager tenemos acceso a un objeto llamado SensorManager que nos permite configurar el tiempo de escucha de los sensores en milisegundos, luego iniciamos un Listener de tipo “Accelerometer” que nos devuelve los datos de la aceleración relativa a la caída libre como marco de referencia (X, Y, Z).

Implementar Sensor de Pasos

Utilizando la librería react-native-sensor-manager tenemos acceso a un objeto llamado SensorManager que nos permite escuchar el evento “StepCounter” el cual nos da accesos a la cantidad de pasos que detecto el dispositivo.

Comunicación entre SE y Android

La aplicación que se ejecuta en la placa Raspberry se conecta mediante socket a un web service para poder comunicarse con la aplicación Android.
Utilizando la librería socket.io en el server y la versión de cliente tanto en la aplicación cono el sistema embebido. Esto nos permite tanto escuchar como recibir eventos.
Para lograrlo se publicó y se expuso el web service a internet mediante una máquina virtual en Google cloud platform.

Links de Interés

Node.js
Socket.IO
Google Cloud

Sistema Embebido

Johnny-Five: The JavaScript Robotics & IoT Platform
opencv4nodejs
face-recognition.js
Raspi-io

Código Fuente

https://github.com/reconocimiento-facial/reconocimiento-facial/tree/master/sistema-embebido

Aplicación Android

React Native · A framework for building native apps using React
react-native-sensor-manager

Código Fuente

https://github.com/reconocimiento-facial/reconocimiento-facial/tree/master/Android