Taller de Proyecto II

2017

Sistema RFID

Proyecto Nº 5

Integrantes

- Basanta, Sofía - 524/1
- Discoli, Tomas - 543/4
- Minig Traverso, Marcelo - 489/7

1. Propuesta Original del Proyecto

Se propuso implementar una API que permita potenciar el desarrollo de futuras aplicaciones que interactúen con RFID, proporcionando la estructura base para las mismas.

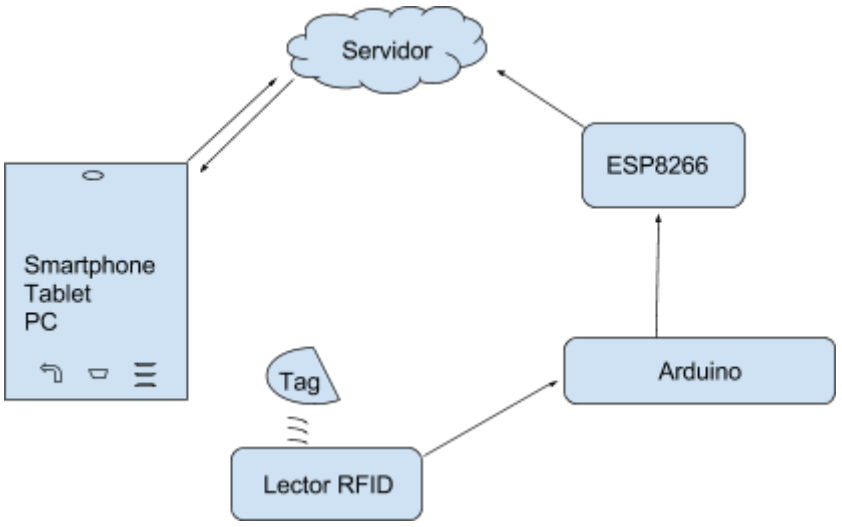
El objetivo es proveer a futuras camadas de alumnos con un sistema autocontenido que posea los endpoints más comunes a la hora de diseñar una plataforma y que utilice identificación por radiofrecuencia como entrada fundamental del sistema. Lateralmente, se desarrollará un sistema que haga uso de esta API para que pueda apreciarse su potencial. Al estar el foco puesto en la implementación de la API como una piedra angular de propósito general, es irrelevante cuál sea el sistema que haga uso de ella. No obstante, se proponen las siguientes alternativas:

- *Sistema de trazabilidad de ganado*: Consiste en poner en las orejas de los animales tags RFID para que al pasar por una manga, un lector pueda identificar al animal y permitir ver a los trabajadores rurales la información concerniente al mismo.
- *Sistema de notificación inmobiliaria:* Se fundamenta en la necesidad de simplificar el cobro y pago de los alquileres por parte de las inmobiliarias. Un inquilino presenta una tarjeta RFID al momento de pagar, el sistema informa el monto que debe cobrarse y cuando el pago es confirmado se envía automáticamente un email de aviso al dueño del inmueble pertinente, generando un historial de pago para futuras referencias.
- *Sistema de validación de tickets en comedores universitarios*: Pretende resolver la congestión que se genera en las sedes más concurridas al momento de retirar el ticket para la vianda. El sistema verifica que un estudiante tenga pagada su ración de comida al momento de presentar el identificador RFID.

Los dispositivos previstos para implementar el proyectos fueron los siguientes:

- **Microcontrolador Arduino Uno R3**
- **Módulo ESP8266**
- **Módulo RFID RC522 1356 mhz**
- **Llaveros o tarjetas RFID**

Flujo de la información:
En el diagrama se muestra la interacción básica de los componentes del sistema y cómo interactúan entre ellos pasándose información.



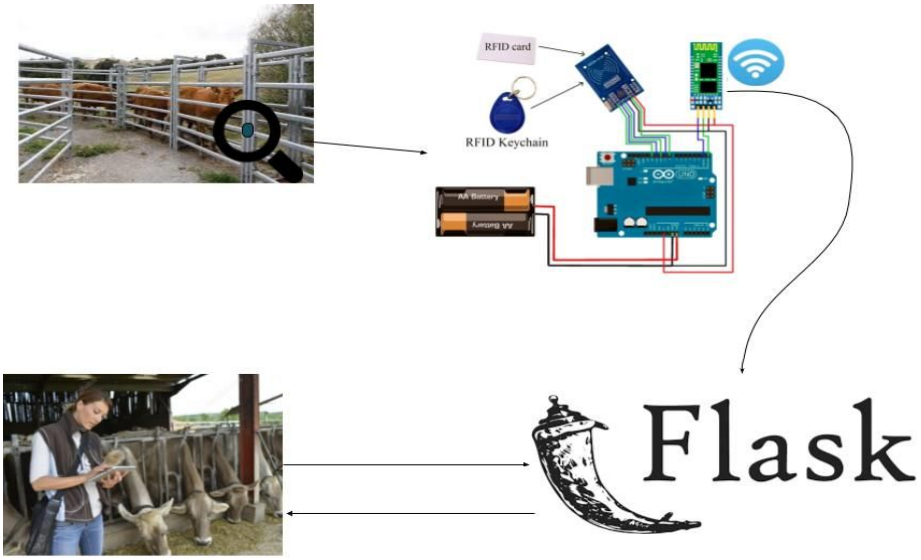
2. Correcciones/Cambios de la Propuesta

Indicadas por la Cátedra

- Ante la propuesta opcional de implementar el sistema con una Raspberry Pi 3 Model B la cátedra nos indicó que la solución debía ser implementada con Arduino.
- Se nos indico que el módulo ESP8266 tenía que estar configurado como Access Point haciendo que el mismo genere un red wifi propia y el sistema funcione dentro de ella. El motivo de esto es facilitar las pruebas, ya que el sistema no depende de una red wifi ajena y al mismo tiempo el sistema está autocontenido.

Definidas por el Avance/Disponibilidad

- Algunas de las pruebas por disponibilidad de materiales se realizan sobre un Arduino Mega en vez de hacerse todo sobre el arduino UNO propuesto.
- Como sistema de ejemplo para el proyecto se eligió el de *trazabilidad del ganado* en el que los animales poseen un tag RFID que los identifica y facilita el acceso a la información del mismo. El siguiente esquema ilustra la idea del sistema planteado.



3. Dispositivos Disponibles

- Esp8266
- Arduino UNO
- Arduino Mega
- Módulo Lector RFID-RC522
- PC con placa de red wifi (servidor)

Todo los dispositivos fueron probados individualmente e integrados al sistema, haciendo que cumplan con su función dentro del mismo.

4. Descripción y Documentación General del Proyecto

Esquemático

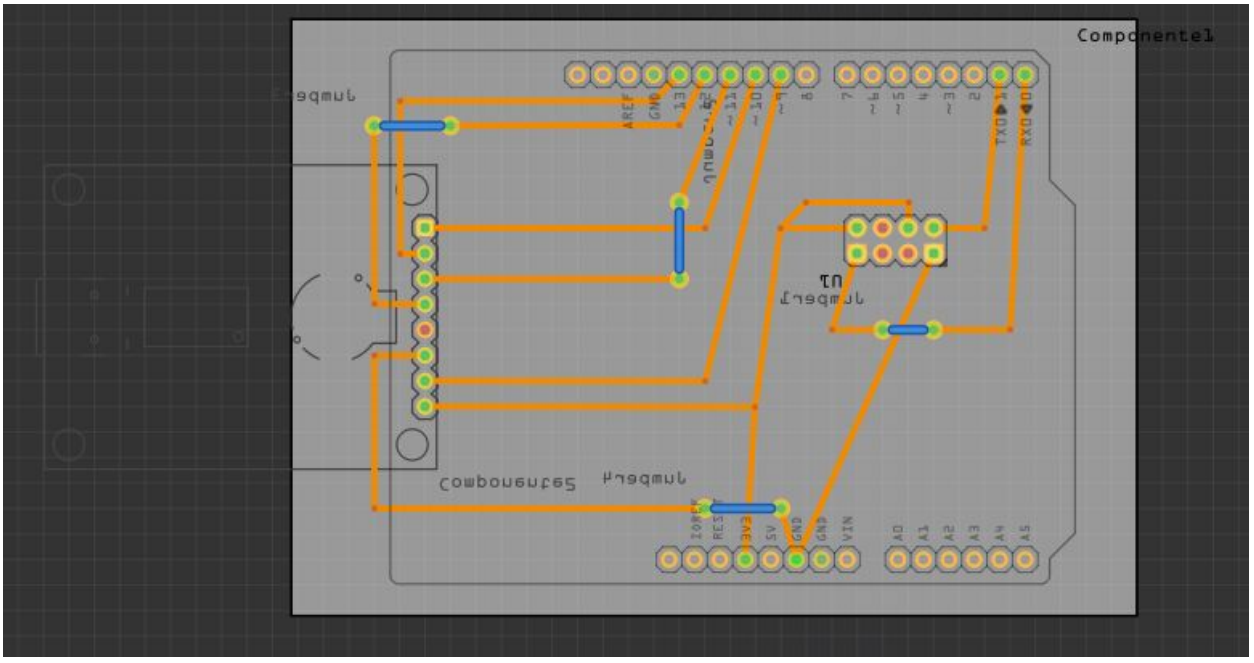
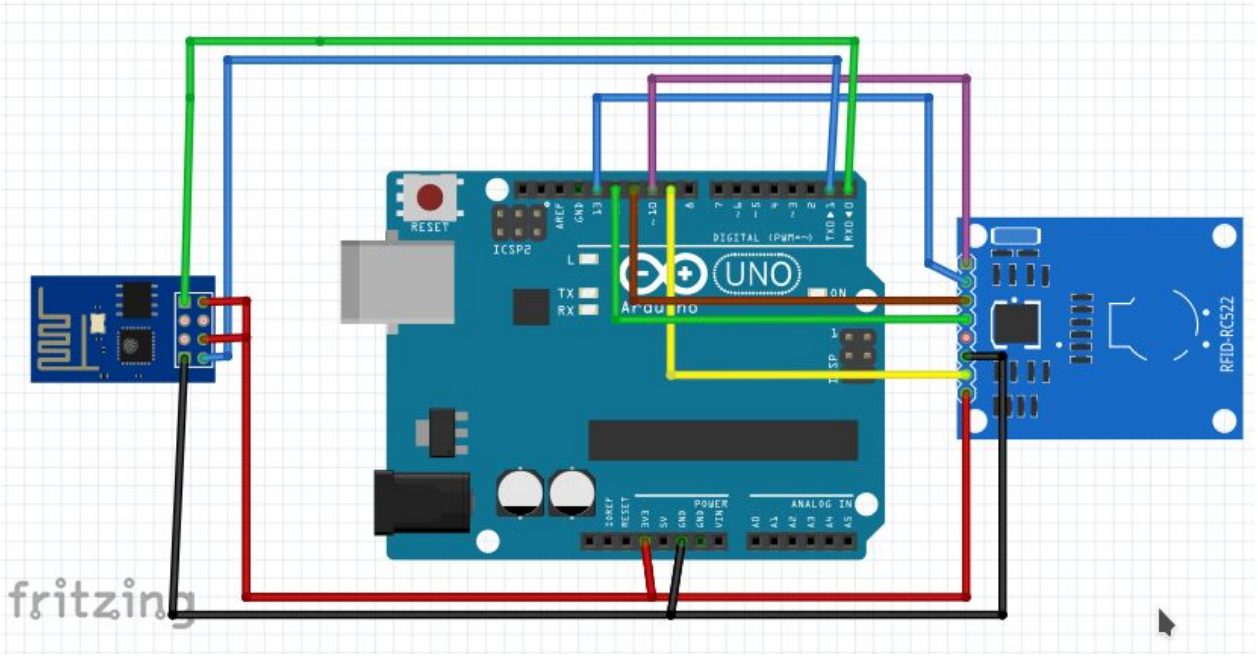


Diagrama de pruebas



Componentes del proyecto

- **Alimentación del dispositivo/placa de desarrollo:** Se estuvo trabajando con la alimentación del puerto serie USB con la PC para poder monitorear el estado del sistema. Las otras alternativas previstas son una fuente de 12v o baterías de no disponerse de conexión a la línea.
- **E/S de la placa Arduino con el exterior excepto PC:** Al Arduino se le conectas dos dispositivos. El primero es la placa lectora RFID que se conecta por SPI y funciona como entrada de información, capturando los IDs de los tags. El otro módulo es el ESP8266 que se conecta mediante el puerto serie (UART), se utiliza para crear la red wifi del sistema al funcionar como AP y para generar peticiones HTTP que se envían al servidor (PC).
- **Comunicaciones de la placa Arduino con la PC:** Mediante el puerto serie USB (UART) se generan logs en consola para poder debuggear el sistema y facilitar el desarrollo. Al mismo tiempo el servidor ubicado en la PC recibe peticiones HTTP por wifi del Arduino.
- **Sistema/interfaz web:** La interfaz del servidor web se puede dividir en dos partes. Una privada que recibe informan de los módulos lectores de RFID, por la cual le llegan los registros de los tag RFID que se leen para ser almacenados. Y una segunda interfaz pública con varias rutas pertinentes al *Sistema de trazabilidad de ganado* que sirven para exponer la información a los usuarios.
- **Infraestructura de software propuesta para la PC:** En la PC se encuentra el servidor, el cual hace uso de la información del sistema. Interconecta el hardware que se encarga de leer los tags RFID con los clientes que pretenden hacer uso de esa información identificatoria. Para guardar dicha información, en la PC también se encuentra una base de datos con el registro de todos los tags que se leyeron, los usuarios asociados a estos y sus datos de utilidad para la aplicación.

Protocolos de comunicación utilizados

- SPI: Es un protocolo de transmisión que permite alcanzar velocidades muy altas y que se diseñó pensando en comunicar un microcontrolador (maestro) con distintos periféricos (esclavos) y que funciona a full dúplex (puede enviar y recibir datos al mismo tiempo).

  SPI provee una solución síncrona, porque utiliza unas líneas diferentes para los datos y el Clock. Este es una señal que indica al que escucha exactamente cuándo leer las líneas de datos, con lo que el problema de pérdida de sincronía se elimina de raíz.

  Por eso mismo, no se necesita pactar la velocidad de transmisión, ya que será el

Clock quien fije la velocidad y puede ser variable a lo largo de la comunicación sin que sea un problema. Aunque por supuesto según el dispositivo habrá un límite de velocidad.

- TCP/IP: El **Protocolo de Control de Transmisión (TCP)** permite a dos anfitriones establecer una conexión e intercambiar datos. Garantiza la entrega de datos, es decir, que los datos no se pierdan durante la transmisión y también garantiza que los paquetes sean entregados en el mismo orden en el cual fueron enviados.
  El **Protocolo de Internet (IP)** utiliza direcciones que son series de cuatro números octetos (byte) con un formato de punto decimal (por ejemplo: 69.5.163.59) para brindar una forma de identificación a todos los dispositivos conectados a la red.
- HTTP: Es un protocolo orientado a transacciones que sigue el esquema petición-respuesta entre un cliente y un servidor. El cliente realiza una petición enviando un mensaje con cierto formato al servidor, y este le responde con la información solicitada. El intercambio nunca es iniciado por el servidor, el cual sólo responde a pedidos de los clientes.
- UART: No es un protocolo de comunicación, sino que es un dispositivo de hardware que facilita la comunicación serie asincrónica, estableciendo el formato de los datos y la velocidad de transmisión, entre otras cosas. Lo incluimos porque es mediante este hardware que se puede realizar la comunicación serie del Arduino con la PC como con el ESP8266.

Software en ejecución

- Software del Arduino: ya se encuentra desarrollado el programa que se carga en el microcontrolador con las funcionalidades de leer tags RFID y enviarlos mediante el ESP8266 a una ruta del servidor por un HTTP/POST. Lo que falta implementar en este aspecto es la etapa de configuración automática del ESP8266 en caso de ser necesario y la forma de identificar la IP del servidor.
- Servidor: actualmente posee todas las rutas necesarias para la interfaz con el hardware como para los clientes del *Sistema de trazabilidad de ganado*, estas están completamente funcionales y accesibles dentro de la red en la que corre el servidor. Lo que falta implementar es la forma de notificar a los clientes de que se registró una nueva lectura RFID al momento en que esta misma sucede.
- Base de datos: es creada y configurada mediante un script. Posee todas las tablas que necesita el sistema para funcionar.

5. Guía de Instalación del Ambiente de Desarrollo y Documentación de Código

Ambiente de desarrollo

Para todo el proyecto se utilizaron herramientas de desarrollo libres, gratuitas y accesibles en distintos sistemas operativos.

La codificación del servidor se hizo en el IDE Atom con el lenguaje de programación Python. Se utilizó el micro framework Flask que se instala con el manejador de paquetes de dicho lenguaje. Para la base de datos se utilizó el SGBD relacional MySql controlado desde consola.

Los programas dedicados al Arduino se desarrollaron en el IDE que esta misma plataforma probe y se hizo uso de la librería MFRC522 que permite interactuar con el lector RFID la cual se puede instalar con el gestor de librerías integrado en el IDE.

Otra opción para desarrollar los programas del Arduino es el paquete PlatformIO-IDE del IDE Atom que se puede instalar desde el gestor de paquetes de este último. Es un entorno de desarrollo para IoT.

Por último para monitorear lo que hace el Arduino por el puerto serie (UART) se utiliza el monitor serie incluido en el IDE de arduino o el programa Minicom que cumple la misma función.

Links para acceder al software mencionado:

- https://atom.io/
- http://flask.pocoo.org/
- https://www.mysql.com/
- https://www.arduino.cc/en/Main/Software
- https://github.com/miguelbalboa/rfid
- http://platformio.org/get-started
- https://help.ubuntu.com/community/Minicom

Para hacer funcionar el sistema solo hay que realizar los siguientes tres pasos:

1. Hay que cargar el programa *final.ino* en el Arduino UNO con el IDE arriba mencionado y mantenerlo encendido con una fuente de alimentación.
2. Crear la base de datos con el script *RFID_BD.sql*.
3. Iniciar el servidor con el comando python app.py.

Documentación de Código

CÓDIGO ARDUINO

Control del ESP8266

```
void reset() {
  esp.println("AT+RST");
  delay(1000);
  if(esp.find("OK") ) Serial.println("Module Reset");
  else Serial.println("Failed");
}
```

Mediante está función se realiza un reset del módulo. No devuelve el módulo a las condiciones de fábrica, sino que lo inicia con la última configuración cargada.

```
void setWifi(){
  esp.println("AT+CWMODE=3");
  delay(1000);
  if(esp.find("OK") ){
    esp.println("AT+CWSAP=\"ESP\",\"password\",1,4");
    if(esp.find("OK") ){
      Serial.println("Creado acces point");
    }
  }else{
    Serial.println("Fallo access point");
    setWifi();
  }
}
```

Mediante este módulo pone al esp como access point, seteando su SSID como “ESP” y su contraseña como “password”. El tercer parámetro(1) indica el canal y el cuarto(4) el modo de encriptación(WPA_WPA2_PSK).

```
void httppost () {
  Serial.println(CIPSTART);//CIPSTART="AT+CIPSTART=\"TCP\",\"192.168.4.2\",8002\r\n"
  esp.println(CIPSTART);
  delay(6000);
  if( esp.find("OK")) {
    Serial.println("Conexión TCP lista");
  }
  delay(1000);
  String postRequest =
    "POST " + uri + " HTTP/1.0\r\n" +
    "Host: " + server + "\r\n" +
```

```
    "Accept: " + "/" + "\r\n" +
    "Content-Length: " + data.length() + "\r\n" +
    "Content-Type: application/x-www-form-urlencoded\r\n" +
    "\r\n" +
    "picc=" + data + "&device=" + device;
  Serial.println(postRequest);
  Serial.println(data);
  String sendCmd = "AT+CIPSEND=";//aca se determina el número de caracteres a enviar
  esp.print(sendCmd);
  esp.println(postRequest.length() );
  if(esp.find(">")) {
    Serial.println("Enviando.."); esp.print(postRequest);
    if( esp.find("SEND OK")) {
      Serial.println("Paquete enviado");
      while (esp.available()) {
        String tmpResp = esp.readString();
        Serial.println(tmpResp);
      }
      // close the connection
      esp.println("AT+CIPCLOSE");
    }
  }
  Serial.println("fin post");
}
```

Esta función es la encargada de hacer el post al servidor de la nueva tarjeta/tag leída. Para ello genera una conexión TCP con la IP del servidor y el puerto sobre el que está escuchando. Por el momento la IP está fija, pero esto se corregirá para la versión final. Se manejan dos alternativas:

- Investigar la posibilidad de asociar una MAC ADDRESS a una IP, quedando la IP fija, pero configurando al ESP para que quede reservada para la MAC del equipo servidor
- Escuchar por nuevas conexiones a la red del ESP, y por cada una de ellas hacer un get a la IP asignada(en una URL definida en el servidor), que devuelva un token que la identifique como servidor, modificando entonces `CIPSTART` con la IP asignada a dicha conexión.

Control del RC522

```
void loop () {
// Buscar nuevos tags
  if ( ! rfid.PICC_IsNewCardPresent())
    return;
```

```
  // Verifico si NUID ya a sido leido
  if ( ! rfid.PICC_ReadCardSerial())
    return;
  Serial.print(F("PICC type: "));
  MFRC522::PICC_Type piccType = rfid.PICC_GetType(rfid.uid.sak);
  Serial.println(rfid.PICC_GetTypeName(piccType));
  // Check is the PICC of Classic MIFARE type
  if (piccType != MFRC522::PICC_TYPE_MIFARE_MINI &&
    piccType != MFRC522::PICC_TYPE_MIFARE_1K &&
    piccType != MFRC522::PICC_TYPE_MIFARE_4K) {
    Serial.println(F("El tag no es MIFARE Classic."));
    return;
  }
  if (rfid.uid.uidByte[0] != nuidPICC[0] ||
    rfid.uid.uidByte[1] != nuidPICC[1] ||
    rfid.uid.uidByte[2] != nuidPICC[2] ||
    rfid.uid.uidByte[3] != nuidPICC[3] ) {
    Serial.println(F("Se detectó una nueva tarjeta."));
    // Guardo NUID en el arreglo nuidPICC
    data="";
    for (byte i = 0; i < 4; i++) {
      nuidPICC[i] = rfid.uid.uidByte[i];
      data+=nuidPICC[i];
    }
    httppost();
    Serial.println(F("El NUID del tag es:"));
    Serial.println(data);
  }
  else
    Serial.println(F("Tag ya leido."));
  // Halt PICC
  rfid.PICC_HaltA();
  // Stop encryption on PCD
  rfid.PCD_StopCrypto1();
}
```

En el lazo loop del arduino se espera por una nueva tarjeta. Cuando el lector RFID detecta una tarjeta, comprueba que no haya sido la leída anteriormente. De cumplirse chequea que el tipo de tag se corresponda con la familia MIFARE Classic. Si se cumplen esas condiciones guarda en la variable data el nuid del tag para que la función httppost pueda persistir en la base de datos del servidor.

Además de los cambios mencionados para determinar la IP del servidor, se planea desarrollar una función que compruebe el estado actual de la configuración del ESP8266, para evitar una reconfiguración innecesaria y aumentar el tiempo de inicio del sistema.

CÓDIGO SERVIDOR

Está desarrollado en el micro framework Flask y hace uso de plantillas HTML, hojas de estilo CSS y scripts JS. También incorpora una serie de funciones que le dan el acceso a los datos de los registros RFID que se encuentran en la BD.

API RFID

```
def RFID_getAllLogs(mysql):
        # crea un cursor a la base de datos
        cur = mysql.connection.cursor()
        # ejecuta la consulta que guarda los datos en la BD
        r = cur.execute("SELECT * FROM logs")
        # persiste los cambio en la DB
        mysql.connection.commit()

        if r > 0:
        # obtengo los datos
        data = cur.fetchall()
        # cierra la coneccion con la DB
        cur.close()
        return data
        else:
        cur.close()
        return 0

def RFID_addLog(mysql, picc, device):
        try:
        cur = mysql.connection.cursor()
        # ejecuta la consulta que guarda los datos en la BD
        cur.execute("INSERT INTO logs(PICC, device) VALUES (%s, %s)",(picc, str(device)))
        # persiste los cambio en la DB
        mysql.connection.commit()
        # cierra la coneccion con la DB
        cur.close()
        return 1
        except Exception as e:
        return 0
```

Estas son dos de las funciones que contiene la API para controlar los datos de los registros RFID. La primera busca en la BD todos los registros (logs) existentes y los devuelve al que invocó a la función. La segunda se encarga de almacenar en la BD un nuevo registro según los datos recibidos por parámetros.

Rutas del SERVIDOR

```
@app.route('/')
def home():

        logs = RFID_Api.RFID_getAllLogs(mysql);

        # renderiza la pagina correspondiente con los parametros que se le pasen
        return render_template('home.html', logs=logs)

@app.route('/newLog', methods = ['POST'])
def newLog():

        log = request.form
        app.logger.info("PICC: " + log["picc"] + "\nDevice: " + log["device"])
        if(RFID_Api.RFID_addLog(mysql, log["picc"], log["device"])):
                app.logger.info("Se cargo el log en la BD")
                return render_template('ok.html')
        else:
                app.logger.info("Hubo un proble al cargar el log en la BD!!")
                return render_template('fail.html')
```

Esta son algunas de las rutas que posee el servidor para brindar información a los clientes. La primera es la ruta raíz, que en este caso se encarga de mostrar un listado de los registros guardados haciendo uso de la API RFID. La otra ruta es por la que los dispositivos Arduino ingresan los nuevos registros al sistema y el servidor se encarga de almacenarlos.

Una vez terminado de desarrollar el sistema se podrá acceder a toda la documentación en el repositorio para entender la completitud del código. Hay que tener en cuenta que gran parte del código del servidor es para darle funcionalidad y sentido al *Sistema de trazabilidad de ganado* y que este no era el principal objetivo del proyecto. Por lo que no se hace foco en cómo el servidor expone la información al cliente.

Base de Datos

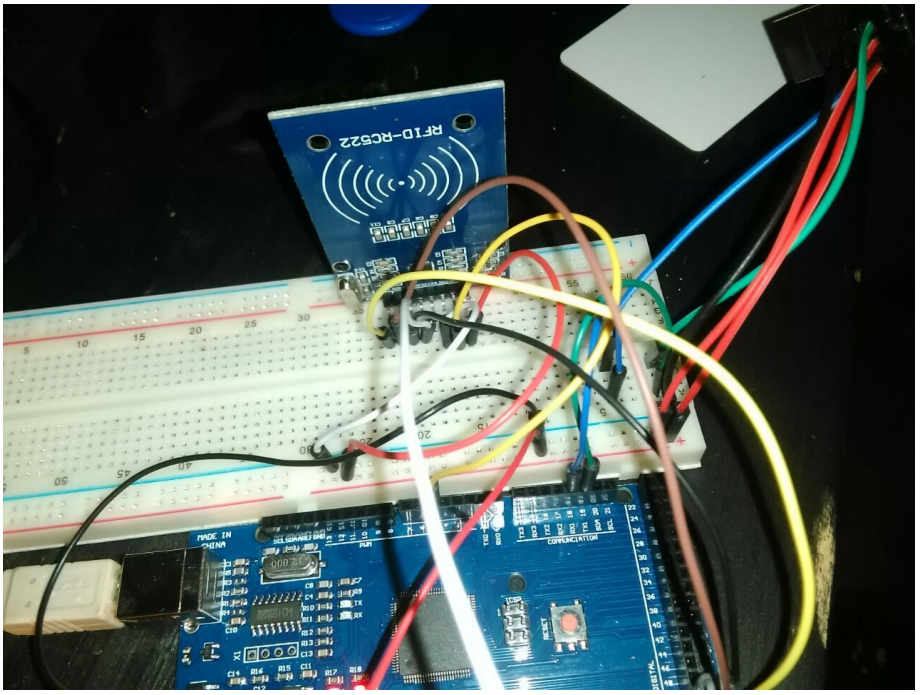
```
CREATE DATABASE RFID_BD;
USE RFID_BD;
--
CREATE TABLE logs(id INT(11) AUTO_INCREMENT PRIMARY KEY, PICC VARCHAR(20), device INT
UNSIGNED, registrated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP);
```

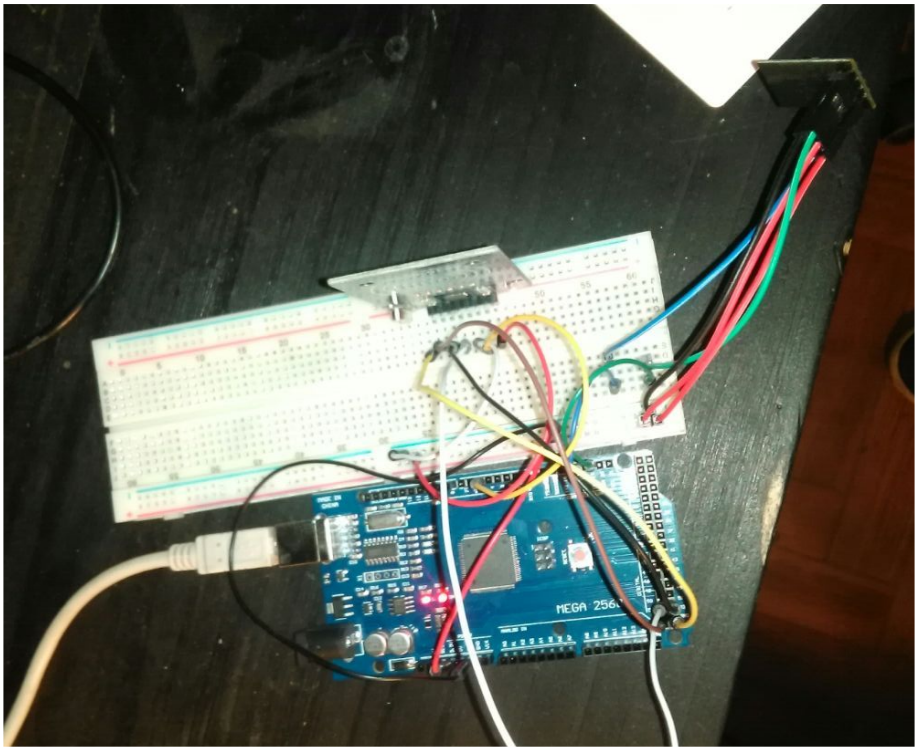
Este script es parte del que crea y configura la BD, en la primera línea se ve cómo es creada y en la última cómo se crea la tabla de registros con sus atributos.

6. Documentación en Formato Gráfico y Video

Sistema en desarrollo

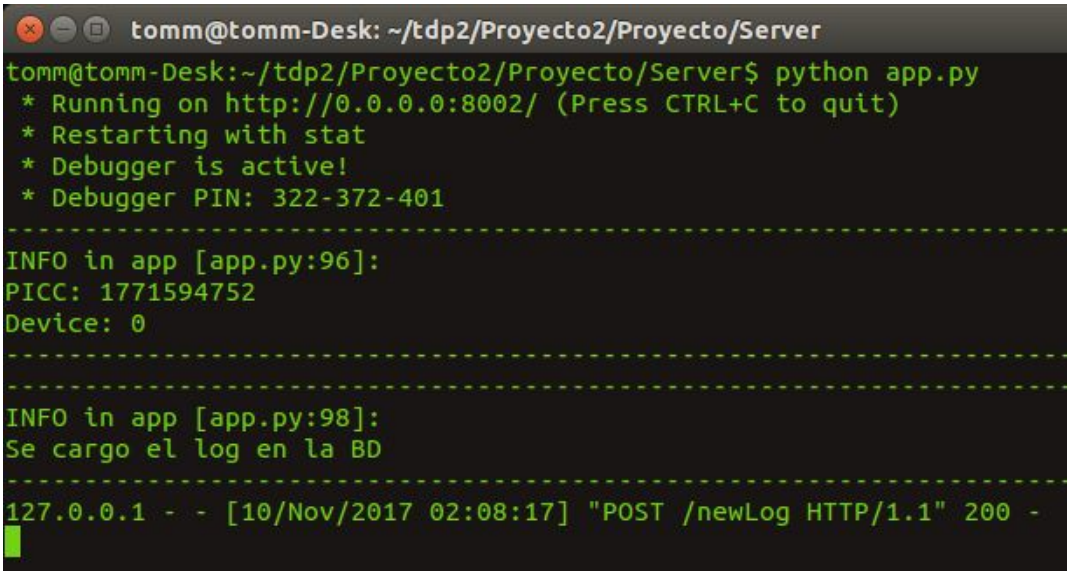
Las siguientes fotos muestran los distintos componentes del sistema conectados entre sí mediante una protoboard.

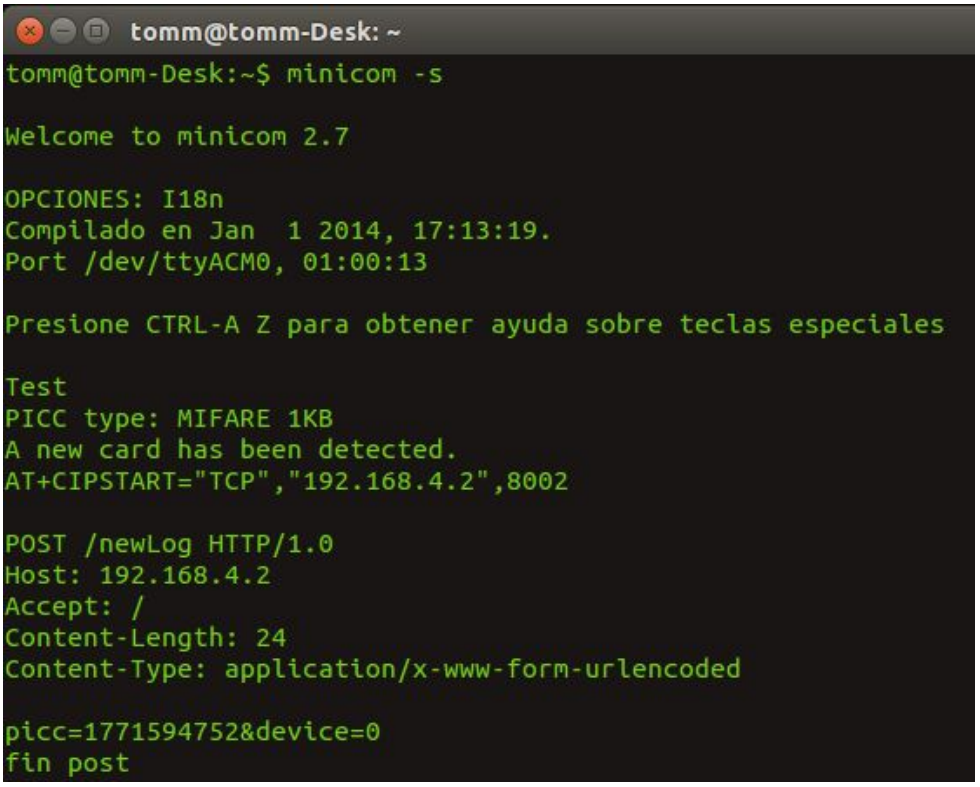




Constancia de la comunicación Arduino-PC

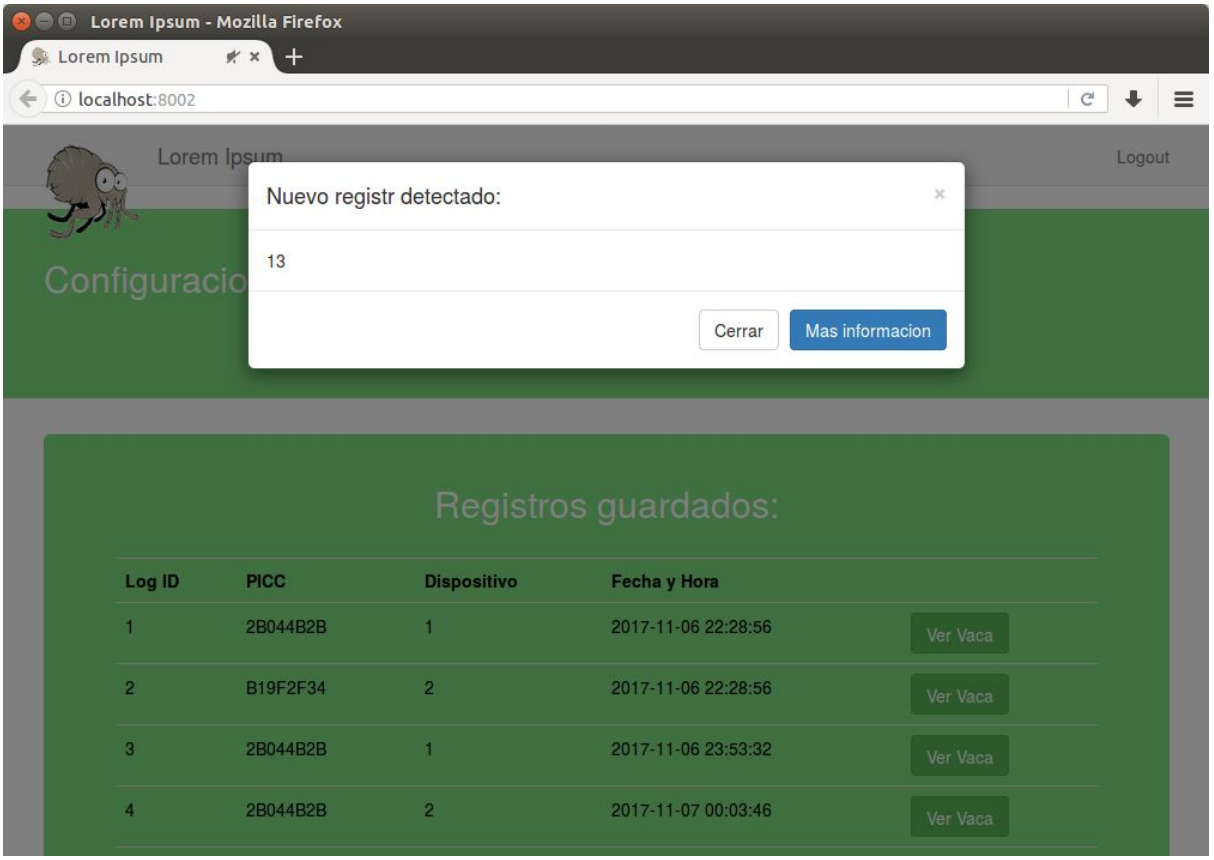
Las siguientes imágenes son los registros de la comunicación establecida entre el Arduino y el servidor en la PC. La primera es el log de servidor que muestra un ingreso “POST /newLog HTTP/1.1” e imprime el número de PICC (ID) y el dispositivo que lo envió. En la segunda se ve la misma informacion, pero formulada por el arduino a través del puerto serie usando Mincom. Se observa que detecta una nueva tarjeta e inicia el envío del paquete HTTP/POST.





Interfaz Web

Las siguiente imagen muestra la página de inicio del servidor, en la que se ve un listado de los registros guardados con su información y un enlace al usuario asociado a ese registro (los usuarios son vacas). Por encima de todo se ve una notificación emergente avisando que ingresó un nuevo registro.



Video

Se puede ver en el video al Arduino ejecutando una petición POST al servidor en el momento en que se pasa una tarjeta RFID por el lector. También se ve cómo es recibida y almacenado en la BD.
Link: https://www.dropbox.com/s/q705j4y8v6dy4mr/RFID_video.mp4?dl=0

De ser posible vamos a subir un video que se vea mejor, ya que no tuvimos tiempo de volver a hacerlo.