Reporte del diseño de la entidad física y del Middleware

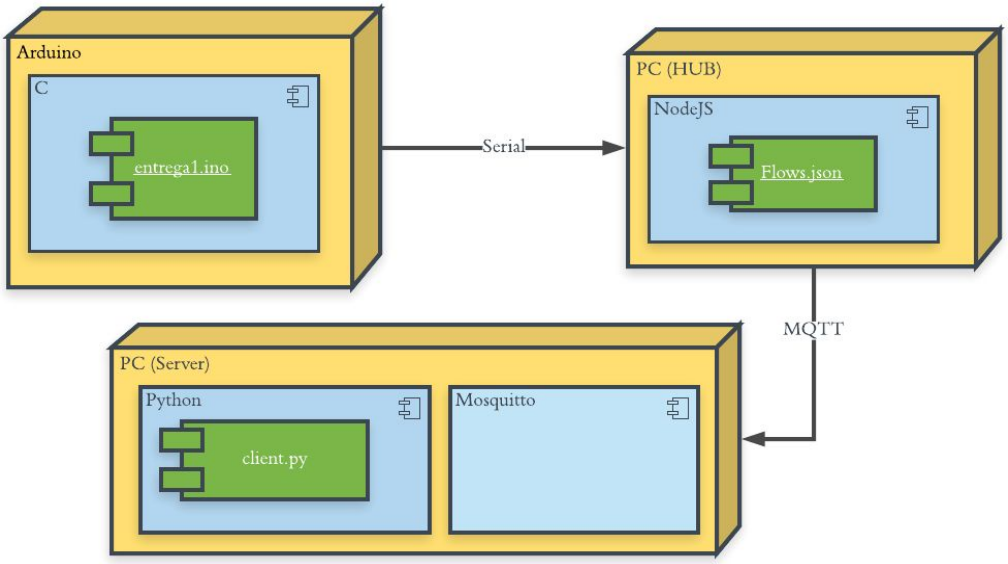
Arquitectura de software

Candax

Mauricio Neira
Stephannie Jimenez Gacha
Juliana Jaime
Laura Valeria Vanegas García
Jeison Mateo Contreras

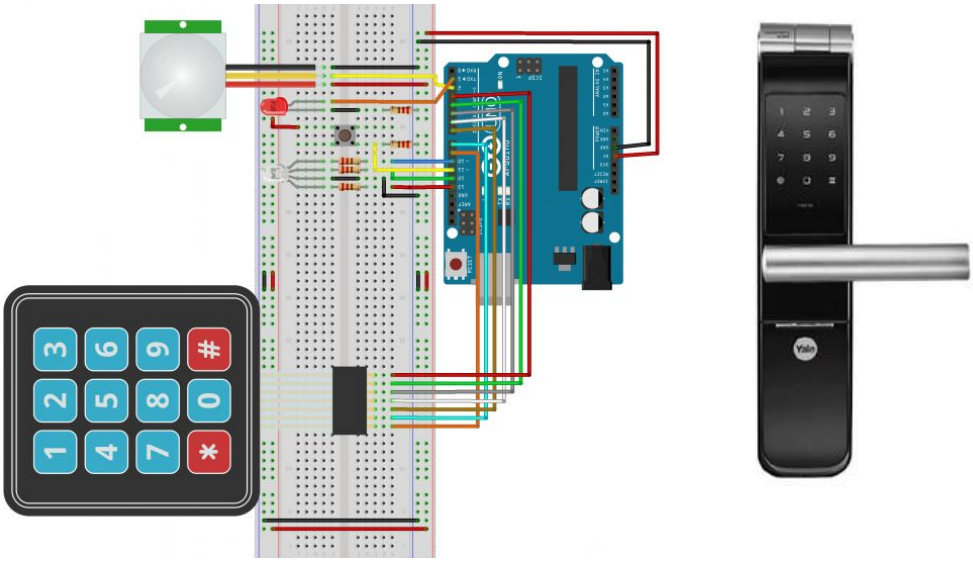
1. El diseño de la arquitectura

Para esta primera parte del proyecto se tiene un diagrama de despliegue en el cual se representa como funciona la arquitectura inicial de este.

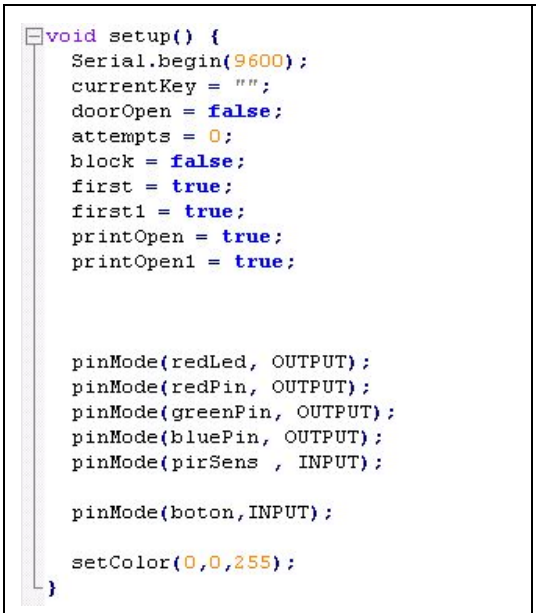


2. La entidad física

El montaje sobre la tarjeta programable se hace de la siguiente manera:



Para el archivo .ino se hace un setup como se muestra a continuación:

	Se establece una clave vacía (“”). Se indica que: 1. La puerta se encuentra cerrada 2. Los intentos fallidos son 0 3. La puerta no se encuentra bloqueada Además se configuran los pines de salida de los leds que indican la situación de la cerradura. De igual forma, se configuran los pines de cada uno de los sensores.

Seguidamente, se verifica constantemente que:

- Se haya abierto la puerta usando la llave
- Se haya abierto la puerta con el uso de la contraseña
- Se digite una clave
- Se hayan excedido el número máximo de intentos al poner la clave
- Se haya cerrado la puerta
- Se haya reiniciado la contraseña

Este programa termina enviando varios de estos datos por serial, entre estos estan:

- *"Door open more than 30s":* Cuando la puerta está abierta por más de 30 segundos.
- *"Door closed":* Cuando la puerta se cierra.
- *"Number of attempts exceeded"*: Cuando se han excedido la cantidad de veces al introducir una contraseña incorrecta.
- *"Door closed":* Cuando se ha cerrado la puerta.
- *"Attempt deleted":* Cuando se ha decidido borrar la contraseña, esto con el fin de no tomar la contraseña como incorrecta.
- *"Door opened!!":* Cuando la puerta se ha abierto, ya sea usando la llave o con la contraseña correcta.
- *"Motion detected!":* Cuando se detecta un movimiento.
- *"Motion ended!":* Cuando el movimiento detectado se detiene.

Todos estos mensajes son enviados, como se evidencia en el diagrama de despliegue, a través de Serial hacia el PC *(HUB)* para ser tratado posteriormente por este en el Middleware.

3. El middleware

a. PC(Hub)

Inicialmente se recibe la información serial dada por el arduino por el nodo serial. Luego de esta información se obtiene el mensaje de la primera línea, dependiendo de este mensaje se envía un mensaje con un tópico en el flujo. Cómo se están modelando alarmas del sistema se decidió discriminar cada una por el nivel de importancia. En el nivel *"High"* se tiene el sensor de movimiento, debido a que si detecta un movimiento sospechoso en la vivienda es la que tendría mayor impacto en la seguridad del hogar. En el nivel *"Medium"* se tiene el número de intentos de ingresar la clave al sistema. Por último, en el nivel *"Low"* se tiene el mensaje de que la puerta ha estado abierta por mas de 30 segundos.

De esta forma se tendría que, si el mensaje es "Number of attempts exceeded", el topico es "medium/house1/permissionDenied". Si el mensaje es "Motion detected!", el tópico es "high/house1/suspiciousMotion". Si el mensaje es "Door open more than 30s", el tópico es "low/house1/doorOpen". Adicionalmente se incluye un payload a este mensaje saliente que es el mensaje original y el id del Hub.

Por otra parte se tiene un nodo inject de timestamp. Este nodo es el que tiene en cuenta el tiempo en el que se generó la alarma. Este se junta en un nodo Merge con el mensaje saliente anterior y este nuevo mensaje conjunto es el que se envía por un nodo MQTT al Broker de Mosquito dependiendo de su tópico respectivo.

b. Cliente

El cliente se desarrolla en Python con la librería paho.mqtt.client. El programa crea una instancia de cliente con el id de éste y lo conecta a mosquitto, indicando la IP y el puerto del servidor. A continuación, se realiza la suscripción a los tópicos definidos en node red. Finalmente en caso de alerta se escribe el mensaje en consola. A continuación, se muestra el código implementado:

```
#!/usr/local/bin/python3
import paho.mqtt.client as mqtt
import time

#Creates the client
client = mqtt.Client("C1")

#Connects to the mosquitto server on port 8083
client.connect("172.24.41.153", port = 8083)

#Subscribes the client to all the topics defined on node-red flow
client.subscribe("high/house1/suspiciousMotion")
client.subscribe("low/house1/doorOpen")
client.subscribe("medium/house1/permissionDenied")

#Definition of on_message
def on_message(client, data, message):
    print("message received: " + str(message.payload.decode("utf-8")))
    print("message topic: " + message.topic)
    print("message qos: " + str(message.qos))

#The client starts reading the message from the topics
client.on_message = on_message

#Start the loop of the client
client.loop_start()
time.sleep(1000)
client.loop_stop()
```