Fire Fighter Car System

Alexandra Ciobotea, Daian Stein, Politehnica University of Timisoara June, 2018

1 Repository

The project history, schematics, diagrams and codebase are contained under the following git repository: **`https://github.com/steindaian/FireFighterCar`**

2 User requirements

1. The system provides information about the environment conditions, as the car is meant to be used as a tool for emergency situations such as CO leak.
2. The system is equiped with humidity, temperature, proximity and CO detection sensors.
3. The tool provides a live video stream which is broadcasted to a controller application.
4. All the environment information are stored in a cloud storage system and can be accessed via the Android application or directly from the WEB; the data is updated once in 10 seconds.

3 System overview

The overview of the system is depicted in figure 1.

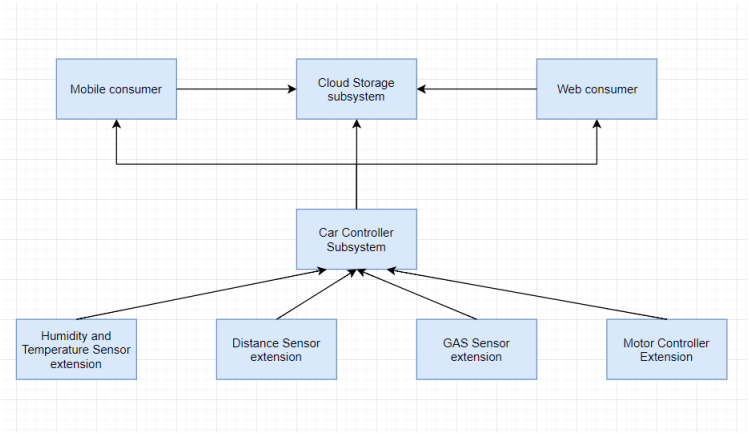

Figure 1: System overview diagram

Base (Car Controller) Subsystem encompasses the measurement functionality, the movement controll of the car and the video stream broadcasting. It has the sole purpose to acquire

information from it's sensors(Humidity,Temperature,CO,Distance-Proximity) and extensions.

The camera(the motion service) together with the engine controller form the Motor Controller Extension.

Cloud Storage Subsystem stores the data pushed by the Base Subsystem. Additionally, it offers a possibility to interpret stored information.

Web consumer provides the video stream depicted by the car's camera. This view is accessed within a Web browser.

Mobile application is in charge with car's movement. It also offer the posibility to analyze data from the Cloud and to visualise the live video stream from the car. The view will be accessed via a specialised application which runs on an Android device.

4 Circuit design

The hardware view of the system is depicted in figure 2.

Raspberry Pi 1 Model B+ provides support for quick prototyping. We will use the one-wire interface it has, but also the implicit possibility of communicating with other devices over the Internet.

For powering the board a 5000mAh power bank is used.

DHT-11 is an one-wire enabled sensor. It is a basic, digital-output relative temperature and humidity sensor.

CO detection sensor is an analogic sensor, which needs an analog-digital convertor(MCP3008) to gather the values properly. For the communication between the convertor and the Pi the SPI protocol was used.

The distance-proximty sensor(an ultrasonic sensor) is a digital component, used for rounding the up-front obstacles.

One of the two engines depicted above is used for speed controlling, while the other one for steering. The battery socket and the motor driver shield offer the strength and precision of the whole car movement.

The system uses Raspberry Pi's Camera v1.

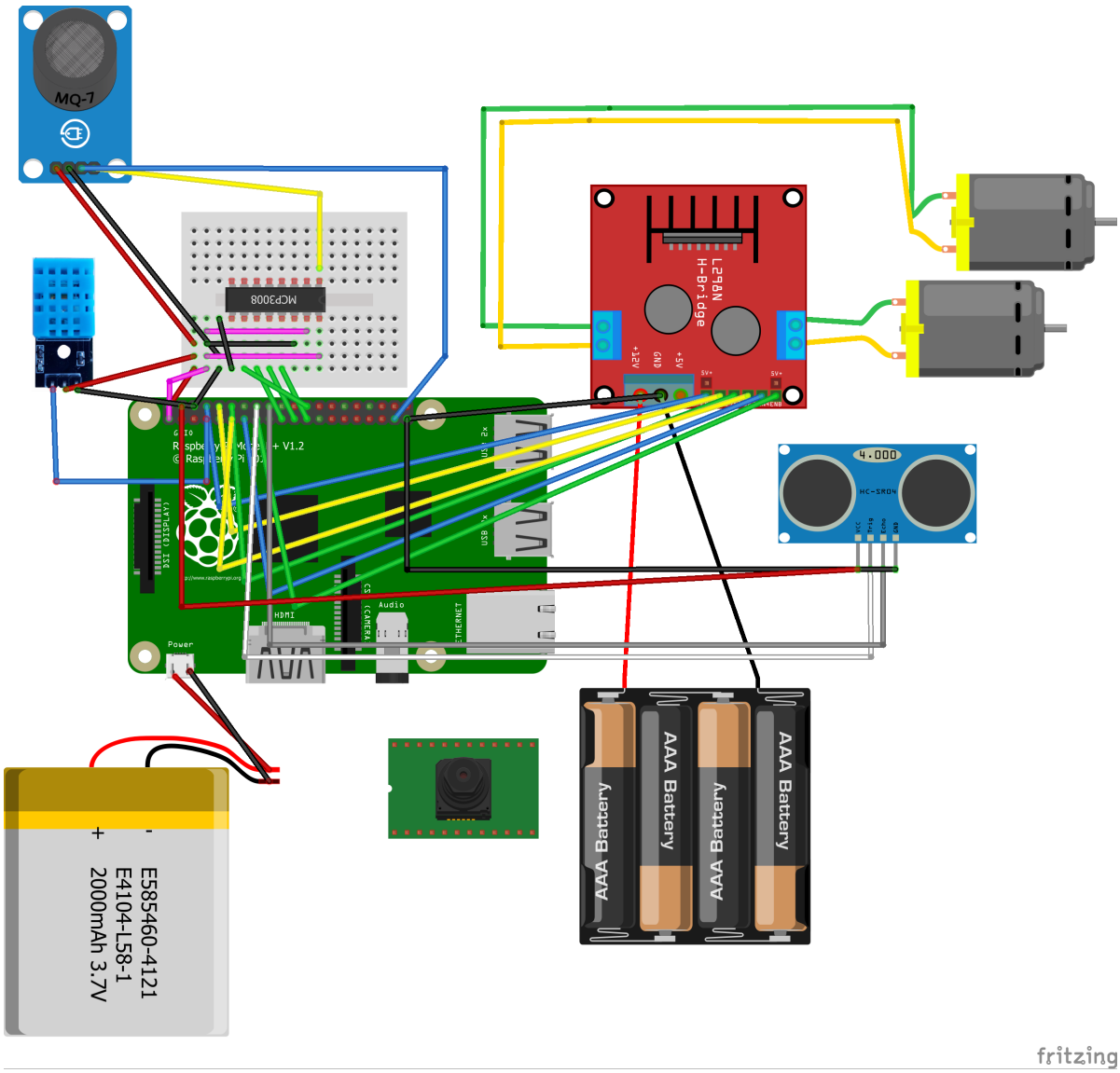

Figure 2: Circuit schematic

5 Software design

The software components and data flow directions are depicted in figure 3. Each of these will be presented in the following subsections.

5.1 Python modules

sensor.py: it retrieves the outside sensor values and then pushes it to Firebase every 10 seconds.

Adafruit_DHT11 library: it provides a quick implementation of one-wire interface communication with the temperature and humidity sensor.

Adafruit_MCP3008 library: it is used for properly reading the CO related sensor values. To-

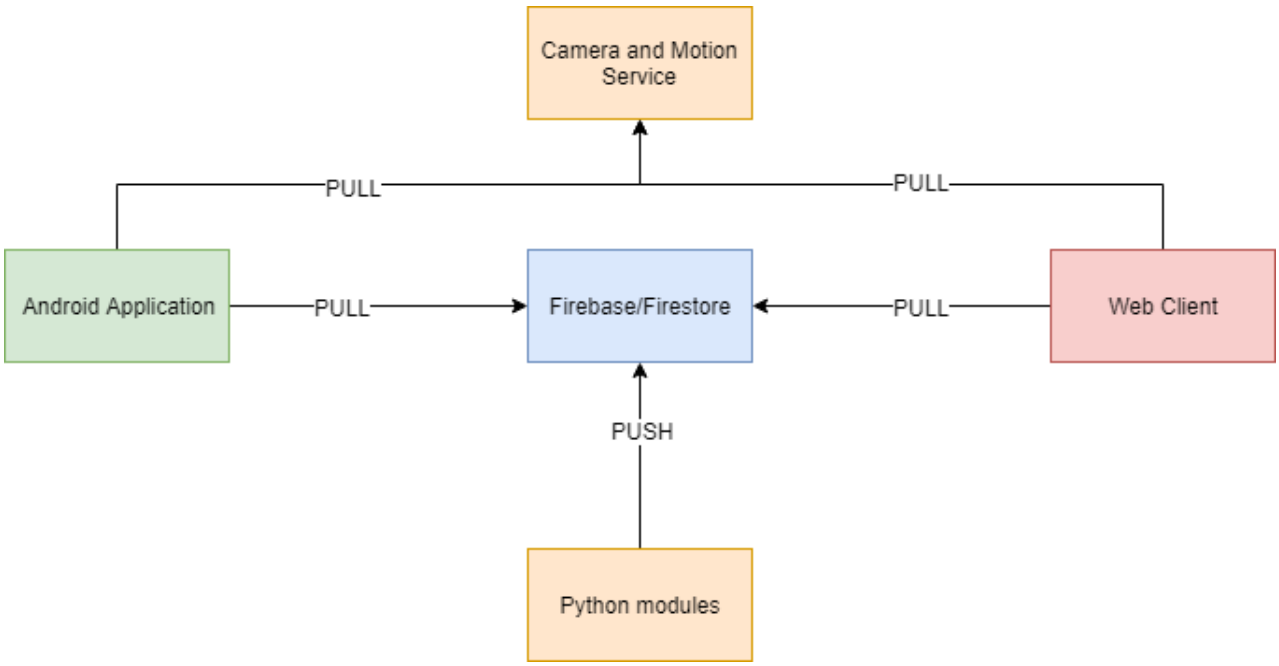

Figure 3: Software entities involved

gether with mq.py library normalize data and saves them for later on.

read_distance function is used for computing the distance from the up-front object, based on simple mathematical equation.

motor.py: it is the submodule responsible with setting the speed and the steering of the car. It uses the RPI.GPIO library for the fine controlling movements(PWM).

main.py: the whole system starts from this script. It uses the above modules and it starts the camera stream service. Also, it creates the server socket responsible for communication with the Controller Application.

5.2 Firebase

Firebase is a PaaS (Platform as a Service) which means it offers developers to a quick list of functionalities supported by a traditional backend. Realtime Database simplifies storing and synchronising data between different devices in realtime using a noSQL database.

The following code section shows the initialization of the Firebase project.

```
# the generated root for your project
FIREBASE_ROOT = 'https://ms-proj.firebaseio.com'
# init Firebase Database instance
firebase = firebase.FirebaseApplication(FIREBASE_ROOT, None)
```

5.3 Android application

The Android application represents the Controller part of the FireFighterCar. The application assumes that the car(Raspberry Pi) is connected to the same network as the application is.

The main features used for constructing this application are:

- A WebView UI component for viewing the video feed streemed from raspberry(+additional camera data)
- Two joystick-like buttons for controlling the steering and the speed of the car; the communication is taking place using socket services
- A TextView with the sensor data from the car(using Firebase Realtime Database)
- Speech to Text translation using SpeechRecognizer service provided by Google
- Concurrent jobs, network-based discovery functions(for searching the Pi's IP) and other mechanism used for building a smooth Controller Application

5.4 WEB client

The Web Client offers a way to debug the video stream, without controlling the motors and cloud retrival information.

6 Results and further work

The current version of the project supports the following functionalities:

- reliable reading of the data sensors
- storing data to Firebase Realtime Database
- a client implementations for retrieving data stored in Firebase Database (Android) and controlling the system
- a way to debug the camera submodule using a simple Web Browser
- a live video stream to the Controller subsystem
- speech recognizer for ease of use(designed for industrial usage)

The following list of extensions and improvements was identified to be supported in the future:

- the system may provide a module for data interpretation.
- remote access for configuring data interval reading

- extend car capabilities of interpreting the video stream(Human recognition)
- improve the hierarchy design of Firebase Database
- a way of centralizing sensor data and outputs frame from the camera(to be used by a supervisor in activity field)
- Making the car both a network device and an access point for better aplicability in real life scenarios

7 References

1. Draw IO [last seen: May 2018], https://www.draw.io/
2. Fritzing [last seen: May 2018], http://fritzing.org/
3. Firebase Database [last seen: May 2018], https://firebase.google.com/docs/database/
4. DHT11Datasheet [last seen: May 2018], https://akizukidenshi.com/download/ds/aosong/DHT11.pdf
5. AdafruitDHT library [last seen: May 2018], https://github.com/adafruit/Adafruit_Python_DHT
6. AdafruitMCP3008 library [last seen: May 2018], https://github.com/adafruit/Adafruit_Python_MCP3008
7. MQ-7 library [last seen: May 2018], https://github.com/tutRPi/Raspberry-Pi-Gas-Sensor-blob/master/mq.py
8. Android Devlopers Guide [last seen: May 2018], https://developer.android.com/guide
9. Motion Camera Service [last seen: May 2018], http://www.lavrsen.dk/
10. Virtual Joystick for Android [last seen: May 2018], https://github.com/controlwear/virtual-joystick-android