VendingHealth

Nadina Ștreangă, George Oprea, Politehnica University of Timisoara May, 2018

1 Repository

The code and project documentation are contained under the following git repository:
`https://github.com/georgeoprea/VendingHealth`

2 User requirements

1. The system must vend snacks to its user based on a calorie based credit system.
2. The device must be able to read the NFC cards provided with the vending machine.
3. The system must allow users to view their credit count in a desktop application.
4. The system must allow users to see the available snacks and their credit cost in a desktop application.
5. The system should allow users to sign up via a web application.
6. The system should allow users to restock or change the snacks in the machine via a web application.
7. The system should be open for client application extensions (mobile application)
8. The system should be open for extension HW extension (adding an NFC writing module so that more than 2 NFC cards could be used with the application)

3 System overview

The overview of the system is depicted in Figure 1.

The base modules (Push button subsystem, RFID Reader subsystem, Ultrasonic Sensors subsystem, Motors subsystem) are the actuators and the sensors of the system. The RFID Reader is able to read a user's card. The push buttons identify the products and receive input from the user in order to select one of the products. The motors start to spin based on the information received from the RPi interaction module. The ultrasonic sensors detect a product has fallen and sends a response to the motors to stop.

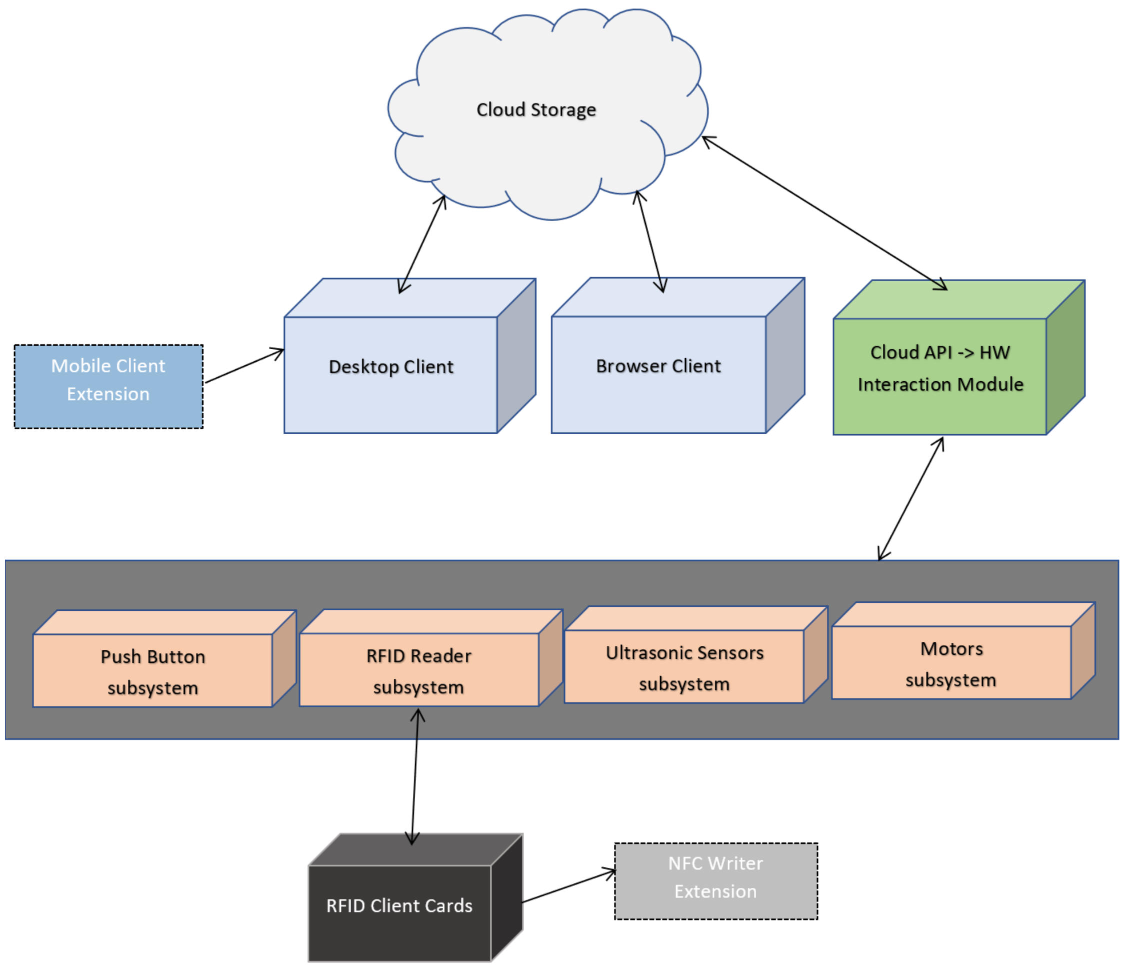

Figure 1: System overview diagram

The HW Interaction Module is a Raspberry Pi that runs a python module. It interacts with the Arduino through a serial interface connection. It also communicates with the Cloud Storage through HTTP requests.

The Cloud Storage Subsystem is a Firebase module that stores user data (credit count, username, password, email), product data (stock, kcal, name, image) and the VendingHealth. client application. It also allows updating users and products.

The Dektop Client provides a UI that allows the user to view his/her credit count and view the products in the vending machine alongside the credits they would conusme.

The Browser Client also provides a UI that connects to the Cloud Storage and allows for an admin to change products in the vending machine and update the cloud storage with the change. It allows a new user to sign up on an RFID card and also provides the means to download the client application.

4 Circuit design

The hardware view of the system and the wiring of the components are depicted in Figure 2.

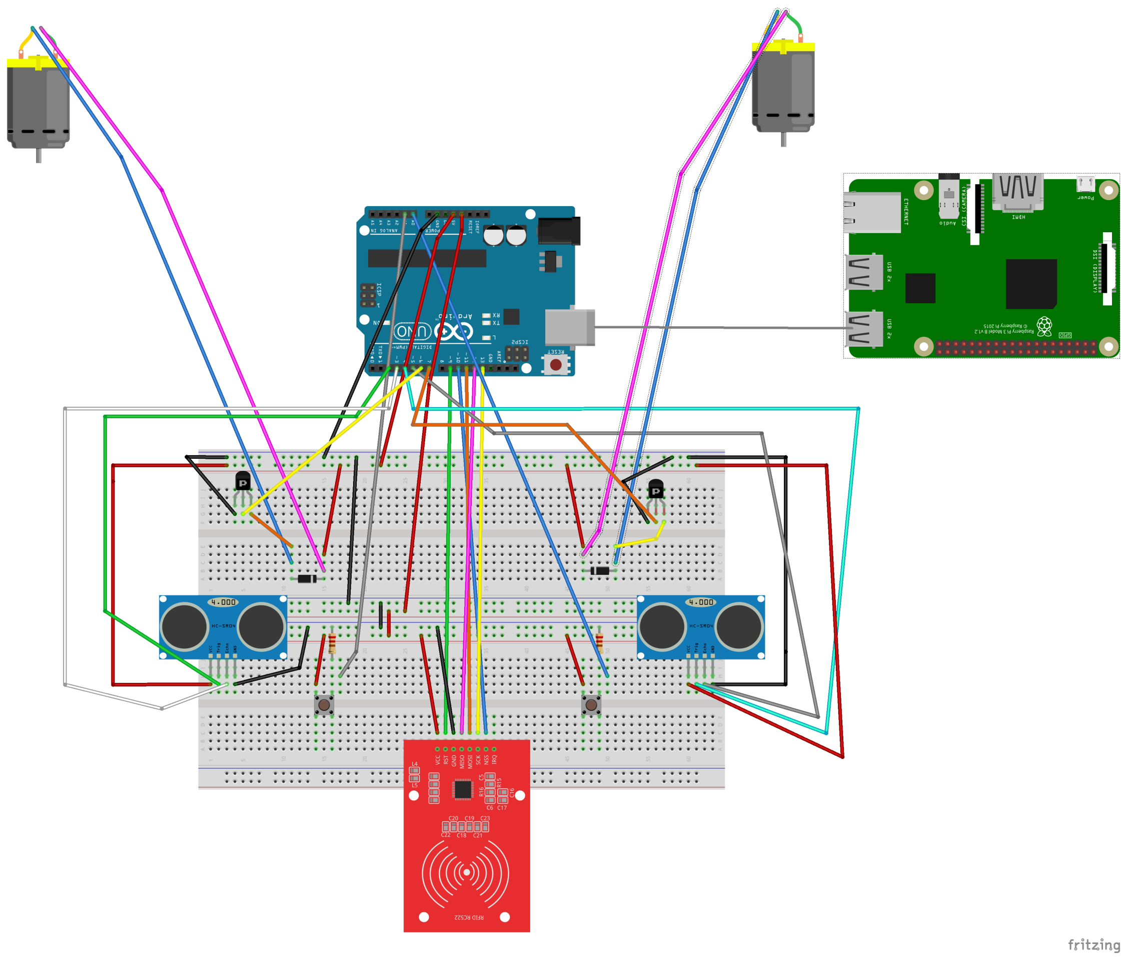

Figure 2: Circuit schematic

Raspberry Pi 3 provides a great interface for connecting to the cloud storage and also to connect to the Arduino. It uses one of its USB ports to connect to the Arduino USB port and receive bits of information. It communicates with the cloud through the internet using HTTP GET/PATCH requests.

Arduino Uno provides power to its components using the 3.3V and 5V outputs. It also connects all sensors and actuators in the same environment and uses both its analog and digital pins to send impulses to all components.

The two HC-SR04 Ultrasonic Sensors use 2x2 of the Arduino's digital outputs for the Trig and Echo pins. The Trig pin triggers a sonic signal and then the Echo pin will transmit the received signal which represents the number of miliseconds the sound wave that was outputted traveled. The sensors are powered at 5V.

RFID MFRC522 is paired with two dedicated RFID cards from which it reads the tag numbers. The RFID reader is powered at 3.3V. It uses the SPI protocol to communicate with the

Arduino. Ports 9 to 13 from Arduino are linked to the MISO, MOSI, SCK, IRQ, RST, NSS pins of the RFID Reader.

The two DC motors are powered at 5 V. They are connected to a diode and a PN2222 transistor. The diode assures that no reverse current will flow from the motor to the Arduino. The transistor allows gaining control of the motor rotation. The transistors are connected to pins 6 and 7, respectively.

The two push buttons are connected to the A0 and A1 pins of the Arduino and are powered at 3.3V.

5 Software design

Provide a walktrough of the most important components/modules/entities or concepts you've implemented in the software section.

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

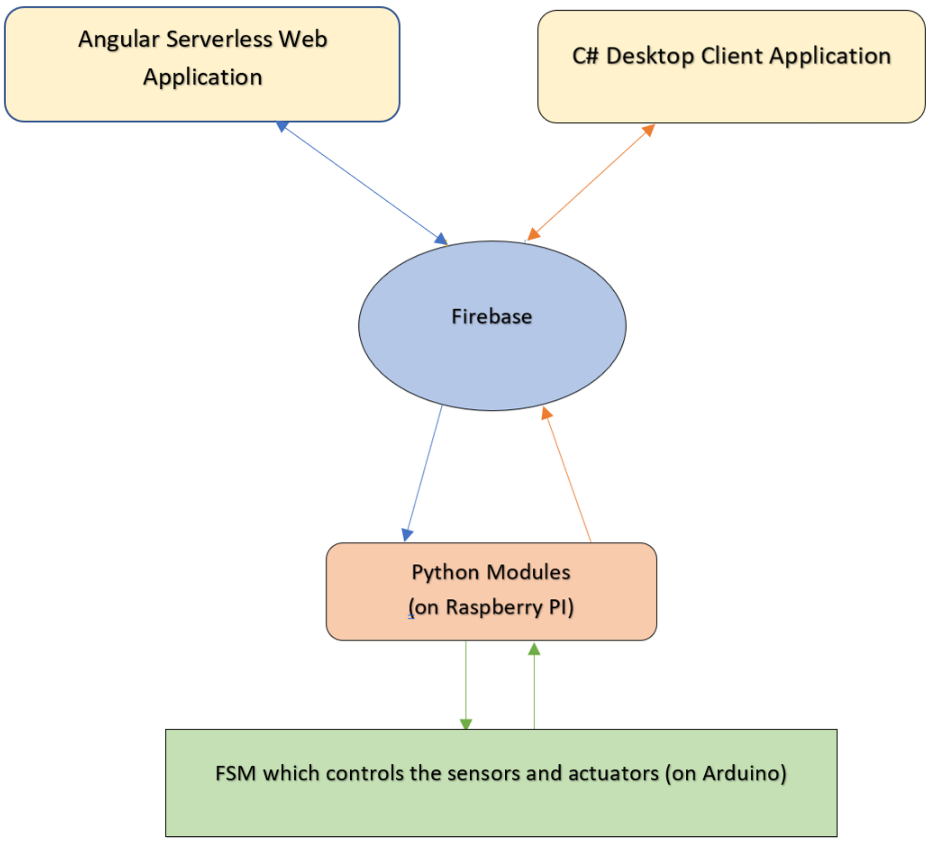
Angular Serverless Web Application
C# Desktop Client Application
Firebase
Python Modules (on Raspberry PI)
FSM which controls the sensors and actuators (on Arduino)

Figure 3: Software entities involved

5.1 Arduino Modules

The main module contains the following main components:

- start_motor and stop_all_motors: simple functions for spinning and stopping motors
- vend: spins a certain motor based on the input from the sensors; if the sensors detect that a product has fallen, the motors will be stopped
- setup: the RFID reader is detected and then initialized; pin modes for all components are set; Serial communication is initialized; SPI is initialized for RFID reader
- loop: implements a state machine with 5 states -> WAIT_FOR_TAG, CARD_CONFIRMATION, PRODUCT_SELECTION, STOCK_AND_BALANCE_CHECK, VENDING; each state implements small parts of the vending cycle, described by the state names

The main module uses the MFRC522 library for RFID handling. It provides an easy interface of connecting and communicating with the RFID reader. It provides functions and macros for setting the pins attached to the reader, getting firmware version and reading the actual tag information from the card.

5.2 Python Modules

You should not concentrate on providing line level descriptions, but rather class/script/module level explanations. Short code comments are strongly advised. Third party libraries should also have a brief description and a reference link.

connectionModule.py:

```
#Arduino Communication
serial_port = '/dev/ttyUSB0'
baud_rate = 9600
ser = serial.Serial(serial_port, baud_rate)
```

```
# root of the project
FIREBASE_ROOT = 'https://vendinghealth-alpha.firebaseio.com'
# init Firebase Database instance
firebase = firebase.FirebaseApplication(FIREBASE_ROOT, None)
```

The functions that operate on the Firebase database:

- updateUserBalance(user, userID, product)
- getUserByID(id)

- getProductByID(id)
- updateStock(product, productID)

An example of an update of a node in the database through the Firebase Python wrapper:

```
def updateStock(product, productID):
  product["stock"] = product["stock"] - 1
  productpath = '/Products/' + ProductID
  result = firebase.patch(userpath, product)
```

The functions that operate on the data rceived from the databse (simple business logic):

- getProductCost(product)
- getBalance(user)
- hasStock(product)
- getProductStock(product)

The main logic of the module is found in an infinite loop which uses the functions above and communicates directly with the Arduino module.

A small fragment of code shows the communication between the Arduino and RPi. The RPi sends a response to the Arduino based on some business logic:

```
if hasMoney(balance, productCost) && hasStock(product):
  ser.write("Y")
  updateStock(product, productID)
else:
  ser.write("N")
```

Firebase library: It provides an interface for Firebase database communication from python, allowing simple retrieval of data from the database.

Serial library: This library acts as a wrapper over the serial interface of the Raspberry Pi and provides functions for reading and writing strings into the serial port (which it marshals and unmarhals into bit streams).

5.3 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 database is divided into two parts: Products and Users. Each of these parts is represented as a node in the database. The Products node identifies the available products from the vending machine, while the Users node maintains the card IDs from the vending machine and the users associated with them.

The cloud storage functionallity is also used to hold the desktop client appllication. It can be downloaded from storage via the web application.

5.4 Desktop Client Application

The Desktop App is developed in C# using Windows Forms. Several other packages are used for JSON handling and HTTP requests handling.

The application is comprised of the following layers: **UI**, **Database Access**, **Data Classes**.

The **UI** has three parts:

- MainApp: Windows Form which shows the user his/her remaining credit count
- ProductsView: Windows Form which allows the user to see the credit cost of the products from the machine
- SignInPage: Windows Form that prompts user to enter his/her user credentials

The **Database Access** layer has two parts:

- ProductInfo: class that connects to Firebase and retreives all products stored in json format
- UserInfo: class that connects to Firebase and checks if a user credentials are valid

The **Data Classes** layer has two classes: Product and User. The JSON data received from the database is deserialized into these objects. They are used by the UI in order to output the necessary information into the view.

5.5 Browser Application

The Browser application is written in Angular and is a serverless SPA.
The following Angular components are part of the project:

- login: allows admin login
- sign-up: allows users to register to a certain card

- home: provides app download, sign-up and login posibilities to the user
- products: allows modification and addition of new products in the vending machine to be represented in the database
- app: entry point of application, provides the routing outlet

The application provides three services: sign-up.service, product-register.service, login.service. They all contain an HttpClient which allows for easy handling of http requests. An example is shown below of a simple get request to firebase.

```
getAdmin() : Observable<User>{
    return this.http.get<User>(this.firebase);
  }
```

6 Results and further work

The current version of the project supports the following functionalities:

- client implementations for retrieving product and user information stored in Firebase Database (Desktop and Web)
- storing new products in the Firebase Database
- allowing an admin to modify products in the Firebase Database
- allowing users to create an account for the application and register a certain card to it
- allowing users to download the application from the browser application
- connection from RPi to Firebase and correct information retrieval and processing
- vending of items independent of Firebase information

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

- include a mobile version of the desktop application
- extend the desktop application to Unix users
- allow extending the number of cards that can be used with the machine
- improve application security (encrypt sensitive user information)
- allow more vending machines in the Firebase database by creating a new node for each one
- allow more snack slots in the machine and database

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. RFID library [last seen: May 2018], `https://github.com/miguelbalboa/rfid`