GIULIA SELLITTO

PROJECT DOCUMENTATION

Università degli Studi di Salerno
Dipartimento di Informatica
Lab of IoT, June 2020

Table of Contents

Introduction

Pets are our best friends since the dawn of time. They are able to put a smile on every face and give unconditional love without asking for anything back. We must love them back and care about them with a shelter, guaranteeing their hygiene and feeding them. Unfortunately, not all people who have a pet are able to be home all day, because of their job. Most of them fill the pet's bowl with food before leaving, so it can eat whenever it's hungry. It would be useful for these people to know whether and when the pet had its meals, in order to keep eyes on its health.

The Project

Smart Pet Bowl is an IoT system in which the pet bowl can automatically refill itself with food at certain scheduled times in the day. The bowl is mounted on a load sensor which can measure its weight. Above the bowl there is a pet food container that can be opened up by a servo motor arm. The user schedules the refill times via an Android app. When the scheduled refill time comes, the bowl weight is measured. If the bowl is not full, the servo motor opens the food container and lets the bowl get filled. All the measured weight data is uploaded onto an online database the app is connected to, in order to provide information to the user about whether, when and how much the pet ate.

Hardware

The hardware schema is presented in the figure below. Each component is described in the following pages.

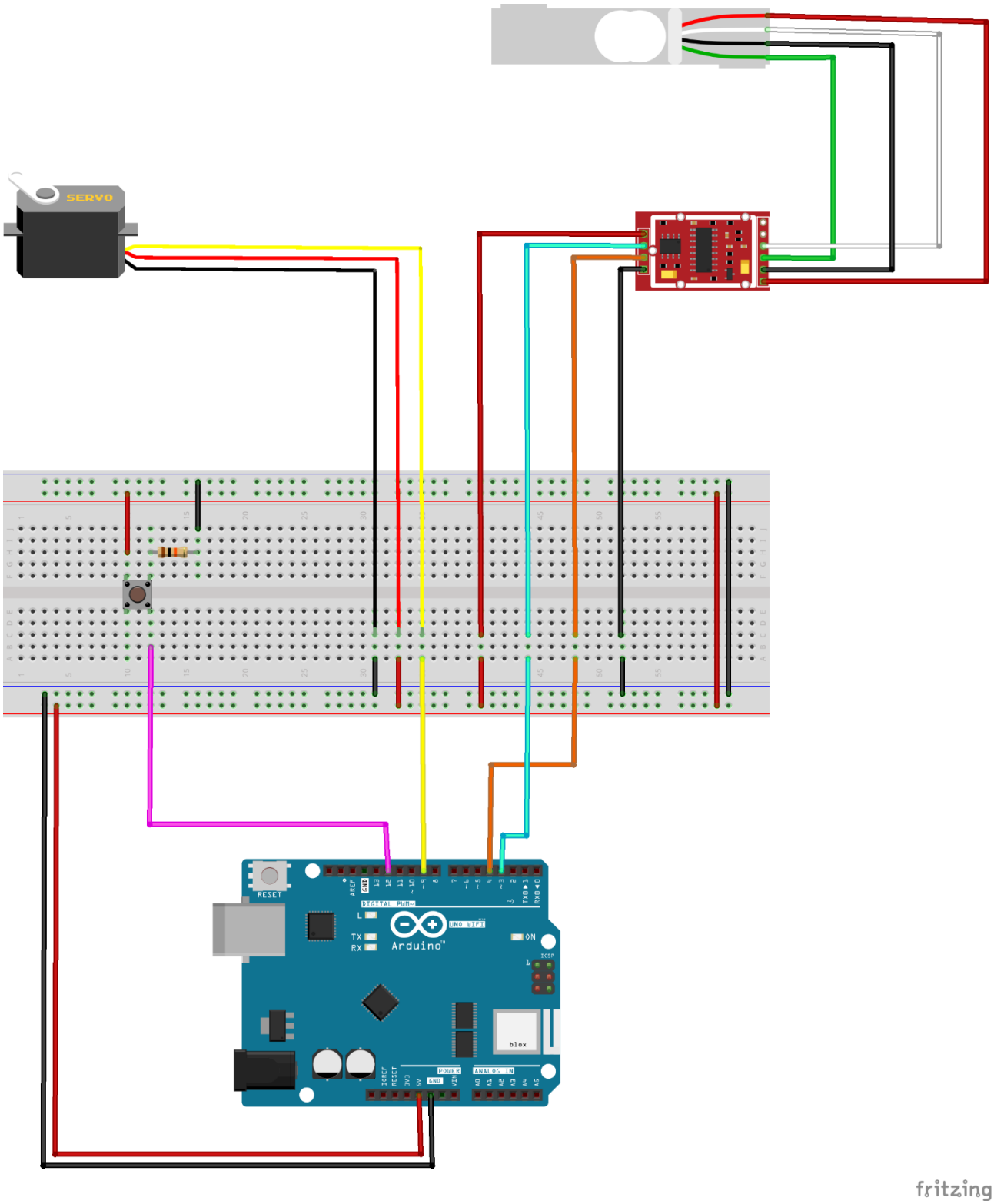


Figure 1: Hardware schema

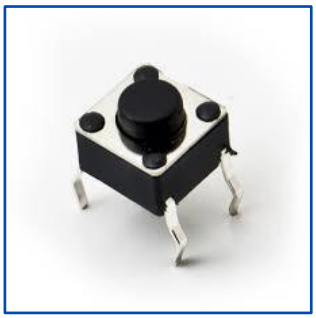
Arduino Controller

The system's core controller is Arduino Uno WiFi REV2. It allows connectivity over WiFi networks and integrates TCP/IP protocol stack functionalities. Connectivity features allow the system to interact with an online database and get and store data in real time. Five of the fourteen digital pins can be used as PWM outputs in order to control analog devices. The servo motor is connected to one of these PWM pins.

Load Sensor

The load sensor consists of a metal bar which can convert the force of the weight sensed in an electrical signal. In order to get the weight data, the controller must communicate to an intermediate module. The HX711 module is able to interpret electrical signal sent by the load sensor and convert them to logical input for the controller board.

Push Button



The push button is a switch used to let the user interact with the controller board. It needs also a 10k ohm pull-down resistor in order to guarantee its functionality. The user can press the button after setting up the bowl calibration, to tell the system that it can start its tasks.

Servo Motor

The servo motor is able to move an arm by implementing a rotation of a certain amount of degrees. Its capabilities allow to change the arm position very quickly, with respect to other motor types. In the project, the arm is provided with a plastic sheet that acts like a door which blocks the pet food from falling into the bowl. The servo motor rotates the arm and the door opens, letting pet food fill the bowl.

Cloud Infrastructure

The project uses *Firebase* as the Cloud infrastructure. It is Google's platform primarily made for mobile apps developing and provides several useful services. The *Realtime Database* allows to easily read and write data as JSON objects. In this project it holds data that describes the weight of the bowl measured by the load sensor at scheduled times. Figure 2 and Figure 3 show the data format.

```
bowl {
        1 : {
                time_h : 10,
                time_m : 30,
                weight : 110.0
            }
        2 : {
                .
                .
```

Figure 2: Bowl data format

```
schedule {
        1 : {
                time_h : 10,
                time_m : 30
            }
        2 : {
                .
                .
```

Figure 3: Schedule data format

The scheduled refill times are written by the user with the Android app and read by the Arduino controller. The bowl weight data is written by the Arduino controller and read by the Android app in order to show the user whether and when the pet had some food. Figure 4 shows the architecture of interactions in the system.

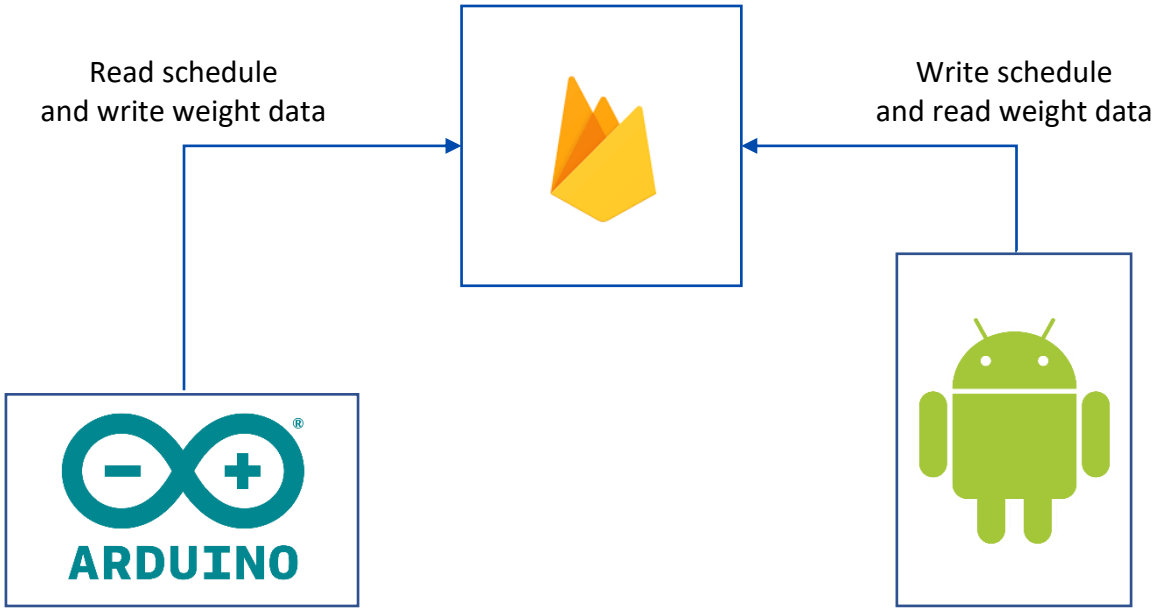


Figure 4: System interactions with the cloud

Arduino Sketch

The Arduino sketch is implemented in such a way that it will ensure a minimum power consumption and memory usage. All variables used are declared *global* in order to avoid dynamic allocation which could lead to full memory and program crash. To ensure power saving, each sensor is deactivated and network connections are closed immediately after use. The overhead caused by switching from on to off state is overcome by the long time in which the system has nothing to do. Keeping sensors and connections up for such a long time would be a waste in terms of power consumption.

Libraries Included

The following libraries are included into the sketch:

- *WiFiNINA.* It enables network connection in Arduino boards that use WiFi as default. It also can manage DNS, a functionality needed in the sketch.
- *WiFiUdp*. It allows the transmission and reception of UDP packets over WiFi.
- *Firebase_Arduino_WiFiNINA*. With this library the sketch can store and retrieve data from a Firebase Realtime Database via WiFi.
- *TimeLib*. It helps time management in the project.
- *HX711*. This library allows the use of the HX711 module for the load sensor.
- *Servo*. It controls the servo motor.

setup() Function

In this function the system is initialized with the following steps:

1. Servo motor is initialized and the arm is set to the closing position.
2. The load sensor is calibrated. The user is asked to first empty the bowl, in order to get the tare weight, then the load sensor can be calibrated by positioning a 100 g load onto the bowl.
3. The user is asked to fill the bowl for the first time.
4. The system connects to WiFi and to the online database for the first time.

loop() Function

In the loop() function these steps are executed:

1. System time is synchronized with the actual time via NTP.
2. Next scheduled refill time is read from the online database.
3. The system will sleep for the remaining time, with all sensors powered down and all connections closed, in order to decrease power consumption to the minimum possible.
4. At the scheduled time, the system awakes.
5. The load sensor measures the current weight of the bowl, which is written to the online database.

6. If the bowl is not full, the servo motor moves its arm to let food fall in the bowl and refill it.
7. The current weight is once again measured and the online database is updated with new data.

Android App

The Android app allows the user to know whether and when the pet had its food and to schedule when the bowl must be refilled automatically. The app was built in MIT App Inventor and it is connected to the Firebase Realtime Database in order to get and store data from it. Figure 5 shows the main purpose of the app: let the user know the pet's meals. Figure 6 shows another screen in the app, where it is possible to schedule the moments in the day when the bowl should be refilled.

Figure 5: App screenshot

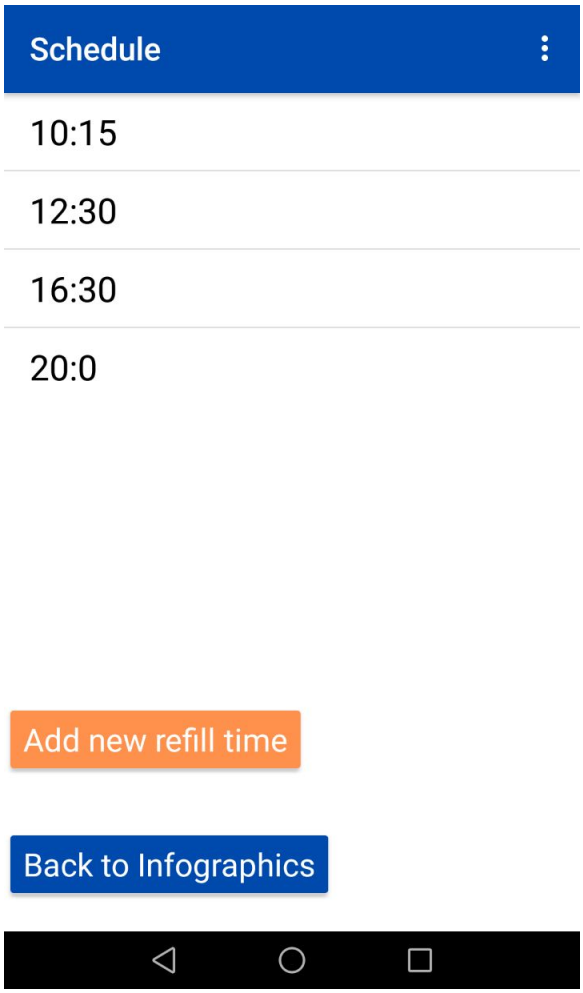


Figure 6: App screenshot

Further Development

The system could be improved by using an LCD screen in order to provide a better UI. Instead of using the serial communication with the console, the user could read the messages on the screen. Also some blinking LEDs showing the system's activity could improve UX.

APPENDIX A: How to replicate the project yourself

You can find the project at this GitHub link. In order to replicate it, you will have to follow these steps:

1. Setup hardware as showed in Figure 1.
2. Setup a Firebase project and choose Realtime Database.
3. Install the following libraries to your Arduino IDE:
    - WiFiNINA (available here)
    - WiFiUdp (available here)
    - Firebase_Arduino_WiFiNINA (available here)
    - TimeLib (available here)
    - HX711 (available here)
    - Servo (available here)
4. Configure source code files *sketch.ino* and *arduino_secrets.h* as described in the files themselves.
5. Open Firebase console and connect a web app to your project. You will obtain a configuration object that must be copied into *chart.html*.
6. Import *android_app.aia* file to MIT App Inventor. Go to *Screen3* and change Firebase configuration in the *Design* view.
7. Import *chart.html* in your MIT App Inventor project.