Introduction to Cyber Systems
Assignment 4 -Implementation of an Internet-of-Things node using the Adafruit Feather Huzzah32 – ESP32 WiFi board and MicroPython

Benjy Jepsen
João Ramiro
Shayan Soltani

December 2018

1 Introduction

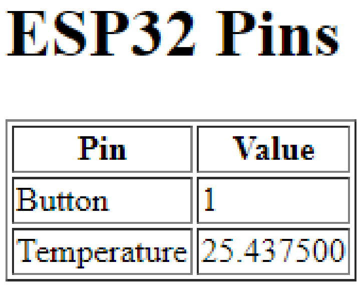
The goal of the assignment was to apply network notions and protocols to control the Huzzah32 board that was used in Assignment 3 in order to get an idea of the way Internet-of-Things devices work. There were 5 tasks to this assignment, outlined below.

2 Task 1

The first assignment was to comment all of the provided `server.py` code. Doing this was useful as it forced us to understand what every single line of code did, which was very valuable for the following excersises.

3 Task 2

In order to print the inputs (which were actually just one input from a button) and the sensors readout (only the temperature sensor) a few changes were done to the provided code. Firstly it was added the pins and functions needed to communicate with the temperature sensor via I^2C and to convert the retrieved value into a float. After that it was just a matter of slightly modifying the provided table to also show the temperature reading. The result is in image 1



ESP32 Pins

Pin	Value
Button	1
Temperature	25.437500

Figure 1: Screenshot of task 2

By refreshing the page, the values would be updated.

4 Task 3

The purpose of this task is to define a simple Web API to interrogate the state of pins and sensors, and to have the information returned in JSON.

The code is implemented on the server side by creating a python function to interpret the TCP lines.

This function is called when a GET is requested through TCP

A list of the pins that are in use is already set up from previous tasks, while a dictionary mapping sensors to their values - including the temperature sensor and a dummy test sensor was created.

The interpreter function takes a given line of TCP code, the list of pins, and the sensor dictionary as input. if `"/pins"` is in a given line, the interpreter iterates through the list of pins and checks if any of the pins is also in the line. The interpreter branches here.

If `"/pins"` is in the line, without a specific pin being identified, the interpreter creates a dictionary mapping pins to their values and then dumps this dictionary as a JSON file. This is then displayed on the webpage.

If `"/pins"` is in the line and and a pin is specified, the interpreter creates a dictionary only mapping that specific pin to its value and dumps it as a JSON file.

The interpreter processes `"/sensors"`, and `"/sensors"` with a defined sensor in a similar manner.

While working through this task - a "Bad Status Line" error kept appearing. This was due to the fact that there was no HTTP/1.1 200 OK being sent from the server. A line containing the acknowledgement to be sent was added manually.

5 Task 4

For this part of the assignment we were asked to write a client side program in Python to monitor the sensor temperature over a period of time. In this case the temperature was being capture 1 time per second during 10 seconds and it is stored in a writable text file. The temperature is requested from the server by sending a GET request to `http://192.168.4.1/Sensors/Temperature`. The server would then return a JSON file of the format `{"Temperature": 24.9375}` and using the json library the values would be parsed and logged into a `csv` (File shown in attachment) file that could be later be converted into a graphic of the temperature over time. In figure 2.

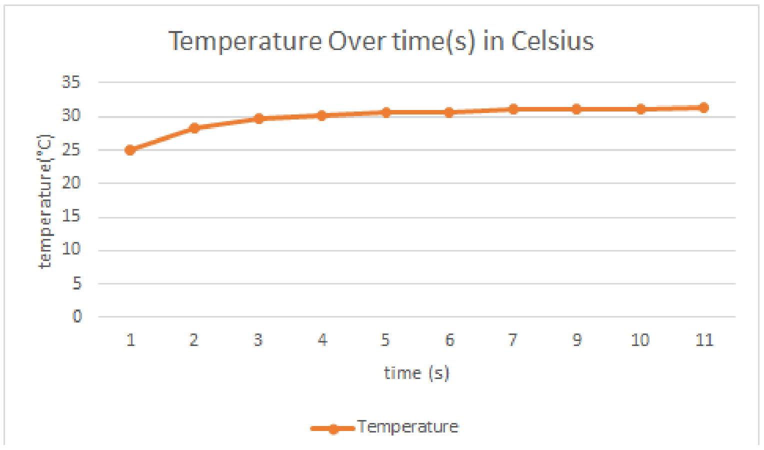


Figure 2: Screenshot of task 2

6 Task 5

The final task wast to make it so that we would also be able to modified the value of the LEDs and Neopixels throughout a client.

The client script was pretty simple:

```
import json
import requests
import time

while True:
    print("Welcome to the danger zone")
    inp = input()
    barred = inp.replace(" ","/")
    send = "http://192.168.4.1/"+barred+"/"
    r = requests.get(send)
```

This script would just pick up whatever the user wrote on the console replace the spaces by slashes and send it to the server. Only the input structure shown in table 1 were valid:

Command	Target	Arg1	Arg2	Arg3	Description	Example
set	Pin(x)	0 or 1			Set pin x to On of Off (1 or 0)	set Pin(33) 0
setnp	y	c1	c2	c3	Set Neopixel y with colors c1 c2 c3	setnp 2 255 0 255

Table 1: Client Inputs Table

On the server, some changes had to be made in order to accept this new type of communication. Pretty much two more if clauses were added to the python function to interpreter function. The first one be to change the neopixels and it would check if the URL contained `/setnp` and it did it would parse whatever came after into a list by splitting by `/` something like `/setnp/1/255/128/3/` , and then it would use the values stored in said list to modify the colors of the specified neopixel. A similar approach was used to change the leds since, first the interpreter would check if the `/set` was cointained in the request and would split by `/` something like `/set/Pin(27)/1/` thus changing the value of that Pin (Led) to On or Off (1 or 0 respectively).

7 Conclusions

In conclusion, this assignment was a good introduction to the "Internet-of Things" as we were able to explore how to create sockets and connect them to ports, how to interpret lines of TCP, how to access data over the internet, how to get data from a sensor through the internet, as well as how to control LEDs and Neopixels over the internet. One comment on the assignment would be that there were no specific guidelines on how to solve the tasks - it was up to us to decide which python dictionary to use to send and receive data (e.g. requests vs urllib/urllib2) Some guidelines on how to implement the tasks could have been useful.