Smart Home Project

Final Report

CS-362, Fall 2017

University of Illinois at Chicago

12/08/2017

Project Members:

Alex Viznytsya

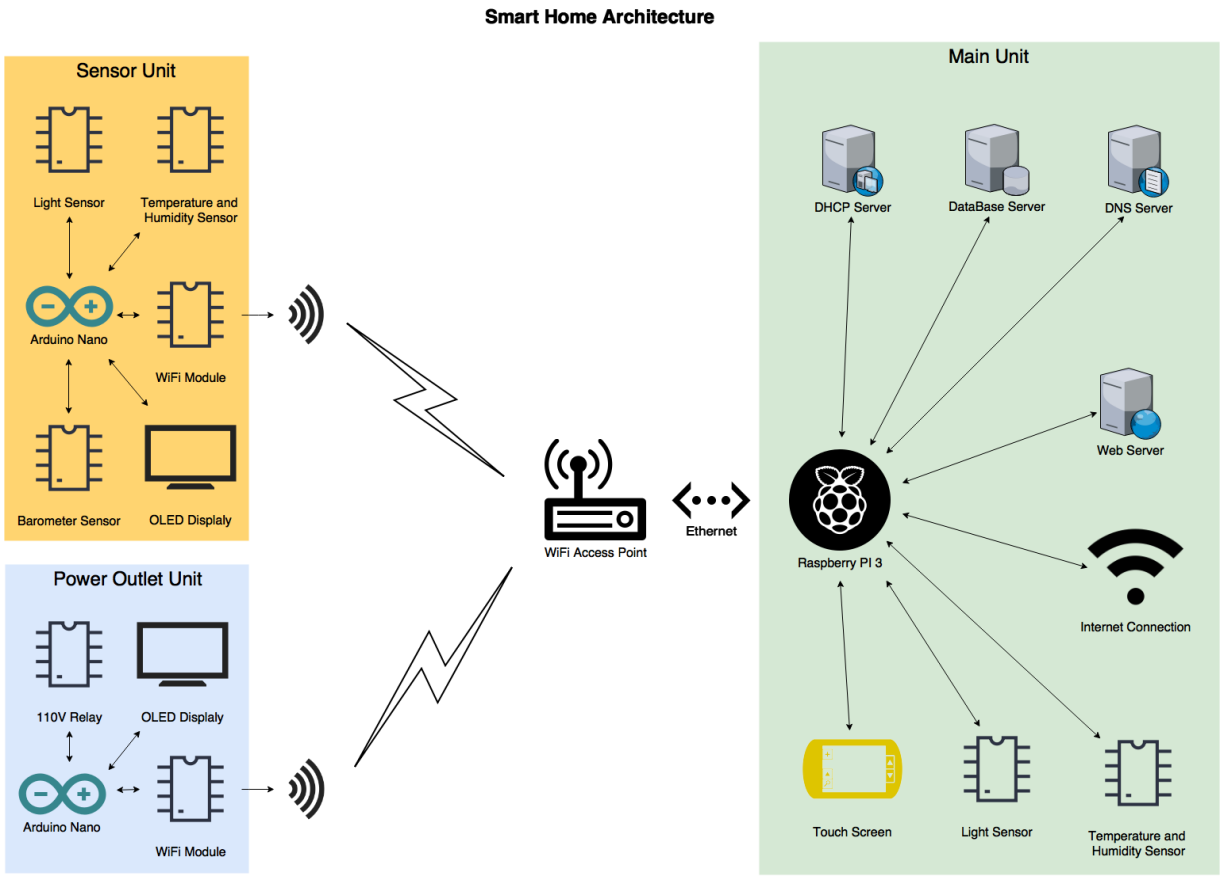
Michal Bochnak

Artur Wojcik

1. Project Overview:

This project of smart home is couple of devices that can control specific power outlets, collect information from home environment (temperature, humidity, light, air pressure) and those devices can communicate with its main unit. Based on data received, main unit will be able to set some criteria to turn on or off power outlets. Moreover, user will be able to control those smart outlets, and get information from sensor units on main unit touch screen or on mobile device using internet browser.

We split this project on three parts, so every project member will do some part of the project in parallel with other members. Artur Wojcik is responsible for Power Outlet Unit, Michal Bochnak is responsible for Sensor Unit, and Alex Viznytsya is responsible for main unit and network connection. Diagram of Smart Home project is below:

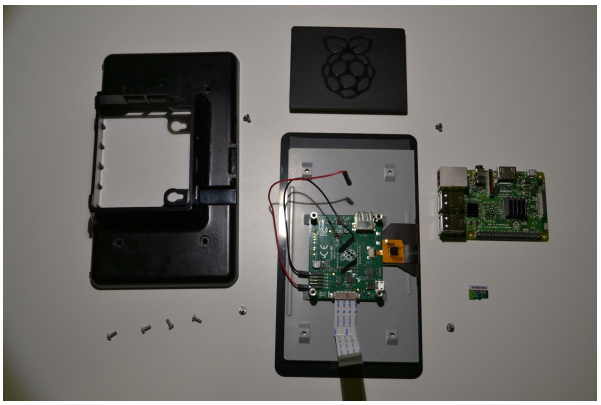


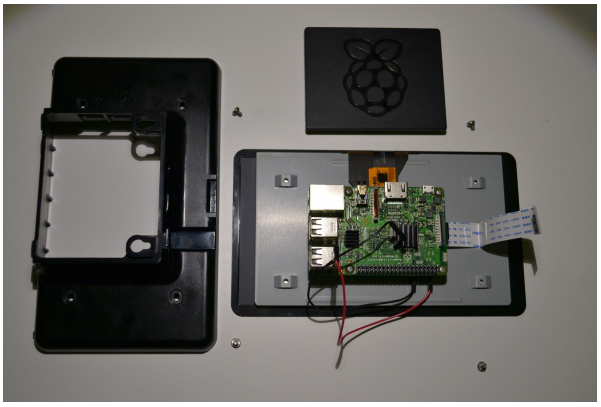
2. Project Development:

2.1. Main Unit:

Assembling Raspberry Pi into Main Unit:

Main Unit is based on Raspberry PI 3 Model B, that has ethernet port and build-in WiFi. I used Raspberry Pi 7" touch screen and case to assembly all these part together into one unit:



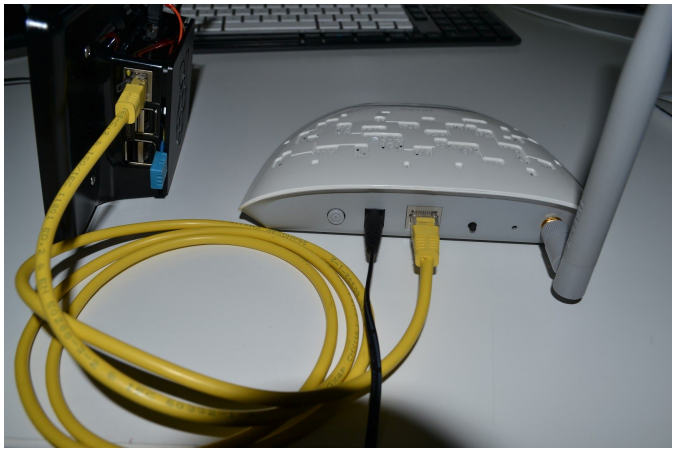


Installation of Raspbian OS:

When Main Unit was assembled, I downloaded Raspbian operating system and installed it on micro SD card using Rufus software. After this, we had the “brain” of our Smart Home project.

Setup of wired and wireless networks:

Next step was creating and configuring wired and wireless networks. We used wireless network as connection to the internet and bridge between Smart Home and outside world. Wired network was used to connect WiFi access point to the Maint Unit and create separate wireless network for all smart devices that we used in this project. So, as access point we used TP-LINK Wireless N, model TL-WA701ND, access point. We connected it to the Maint Unit using ethernet cable. Then, access point was configured to have SSID as “Smart Home” with password “smarthome”, IP 192.168.254.254 / 24, and do not use build in DHCP server.



During network setup I encountered issue that wired network connection had higher priority than wireless connection, and when wireless access point was connected to Raspberry Pi via ethernet cable, it did not have internet connection via WiFi. To fix this I manually added wireless adapter `wlan0` and static `eth0` configurations to `/etc/dhcpd.conf`:

```
interface wlan0
metric 200

interface eth0
static ip_address=192.168.254.1/24
static routers=192.168.254.1
static domain_name_servers=192.168.254.1
metric 300
```

Installation of DNS/DHCP server:

After unit reboot two networks began to work as planned. Next step was installing and configuring DNS and DHCP servers. This gave us ability to use simple smarthome.local domain name inside of “Smart Home” network and automatically assign ip addressed to all connected devices and “smart” units. I used dnsmasq DNS/DHCP server because it uses small amount of system resources and available for Raspberry Pi platform. So, to install it i used command: `sudo apt-get install dnsmasq.` Then inside of folder `/etc/dnsmasq.d` I created file `smarthome.dns` and add configuration below:

```
domain-needed
bogus-priv
local=/local/
domain=local
expand-hosts
dhcp-range=192.168.254.10,192.168.254.100,255.255.255.0,72h
dhcp-option=option:router,192.168.254.1
dhcp-option=252,"\n"
```

After restarting DNS/DHCP server our two networks worked well and we could added and track if our “smart” units could connect to our wireless network.

Installation of LAMP servers:

Next, I began of installation of LAMP (Linux: Apache, MySQl, PHP) applications. To install them I used commands below:

Apache2 Web Server:

```
sudo apt-get install apache2
```

PHP 7:

```
sudo deb http://mirrordirector.raspbian.org/raspbian/ stretch
main contrib non-free rpi
```

```
apt-get install -t stretch -y php7.0 php7.0-cli php7.0-opcache
php7.0-mbstring php7.0-curl php7.0-xml php7.0-gd php7.0-mysql
```

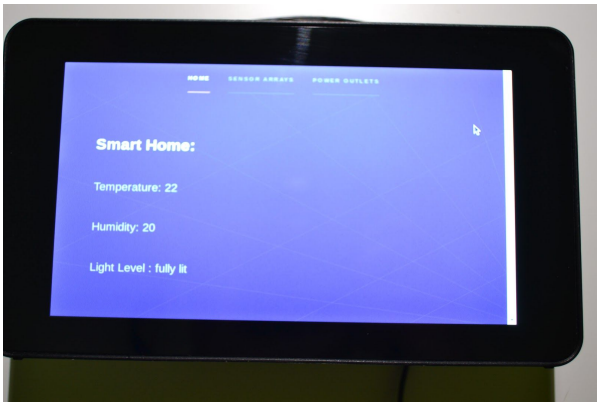
MariaDB MySQL Server and PHPMyAdmin:

```
sudo sudo apt-get install mysql-server
sudo mysql_secure_installation
sudo apt-get install phpmyadmin
```

After these commands and following installation instruction I got working LAMP server stack. Then using PHPMyAdmin I created new database “smarthome” and new user “smarthome” and password “smart” with granted all privileges for that specific database. Moreover, I created two php scripts that let both Sensor Unit and Power Unit send and check data from database, so we could check if network connections and LAMP works correctly.

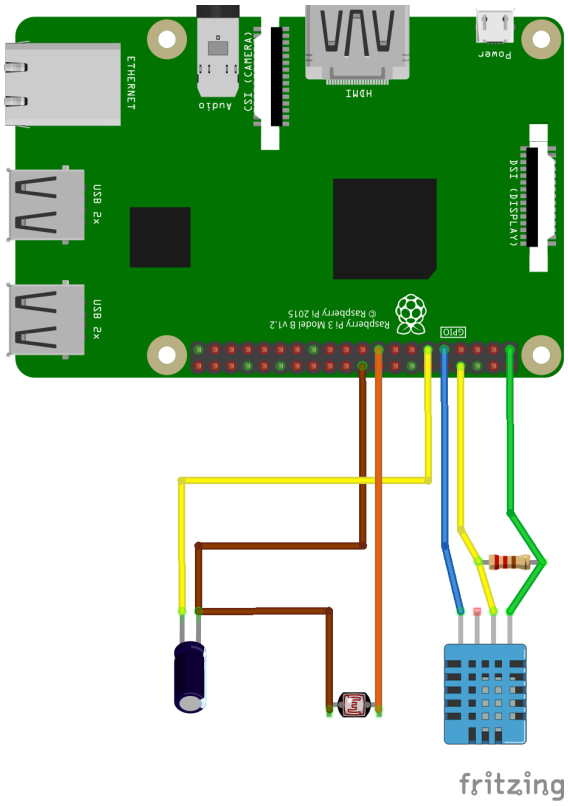
Main Unit web interface:

To create web interface for Maint Unit I used free HTML5 template named “Hyperspace” created by “HTML5 UP”. I removed all their web layout and created my own layout that will fit and looks nice on Raspberry Pi touch screen. Every unit got its own tab and animation between tabs transitions. Also I wrapped HTML code inside PHP, so Main Unit could access database and prints updated sensors readings and power unit status every second.



Sensors to Raspberry Pi:

The last step in my Main Unit development was add light and temperature/humidity sensors to Raspberry Pi. When I wanted to connect light resistor to Raspberry Pi I realized that Raspberry Pi does not have any analog inputs pins. I did research and found that charging and discharging capacitor can be used as analog to digital converter. So, I used that technique in combination with Python, and I got good readings. DHT11 was not a problem, I used Python to get readings from it. Finally, PHP script was used to run and get data from python scripts and print that information on Raspberry Pi display.

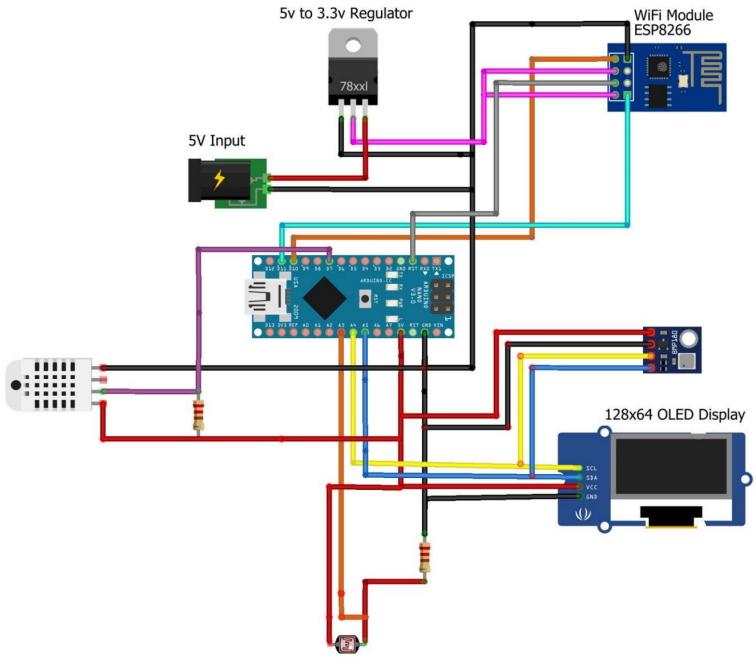


2.2 Sensor Unit:

Sensor Unit development was split up into stages. First, every component was assembled separately and wired to arduino uno to make sure component is working properly. After that all of the components were added one by one to the arduino uno. With every component added the program that was controlling the board was updated by the functionality for the given sensor. Order of adding the components does not matter in fact, however it is very important that after adding each of the component the

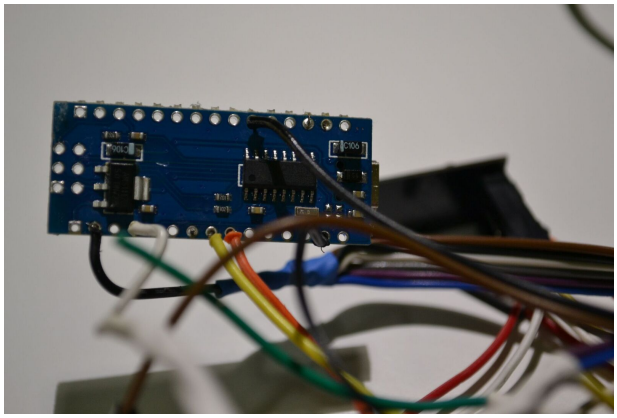
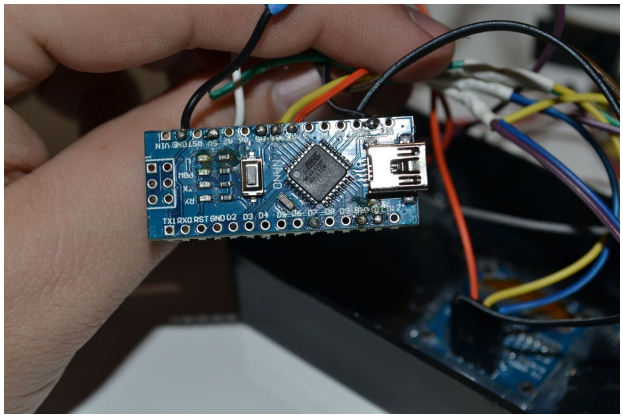
unit must be tested before proceeding to adding next component. That step is critical and should not be omitted in order to minimize possible issues with the system. Once unit was assembled and tested using arduino uno board next step was to solder components to the arduino nano board to create one unified system. Process of soldering was similar to the assembling the unit. One component at the time was soldered and verified if soldering was correct before continuing. Process described applies for all of the components used to build the unit. Step by step development is shown below.

Wiring Diagram:

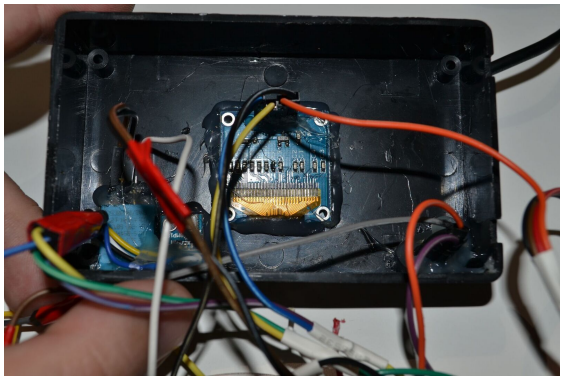


Arduino Nano:

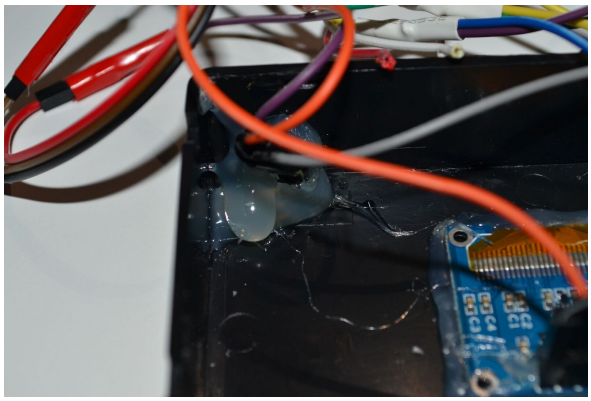
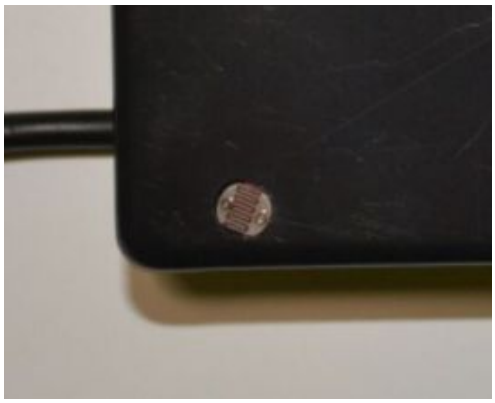
Once all the components were set and tested on the arduino uno board they were soldered to the nano board and installed inside the box. Nano board shown below.

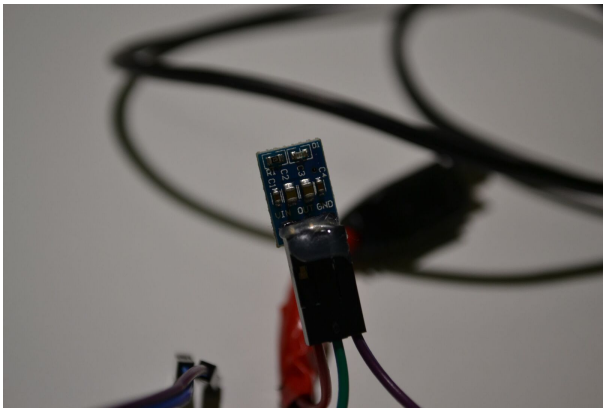
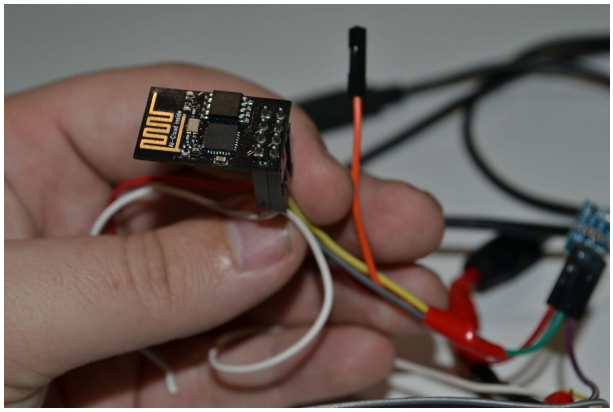
128X64 OLED Display: Development of the Sensor unit was started with the screen. Already installed at the picture below.



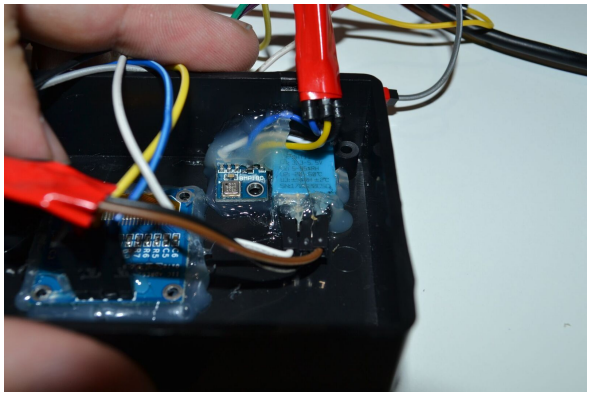

Photocell: Next photocell was installed. At the picture below attached by using plastic glue.

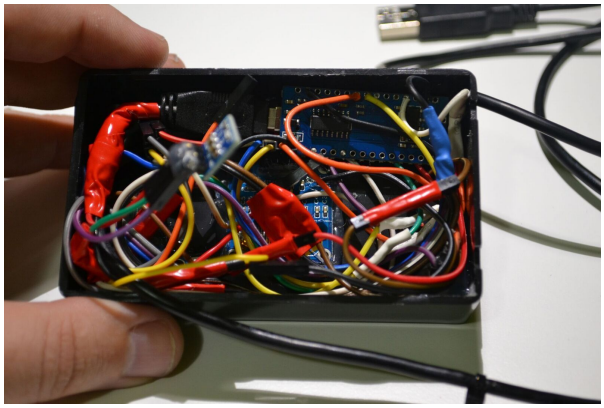
WiFi + Voltage regulator: Following was WiFi installed along with voltage regulator. Arduino nano was not able to supply all the devices connected we supplied external 5V power supply. Since WiFi works on the 3.3V, voltage regulator was needed to convert from 5V to 3.3V.

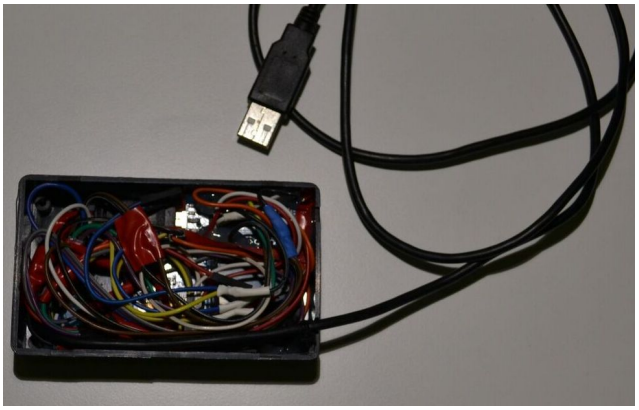



Barometer / Temperature and Humidity sensor: Next step was to install barometer (on the left in the corner) and temperature and humidity sensor (on the right in the corner).

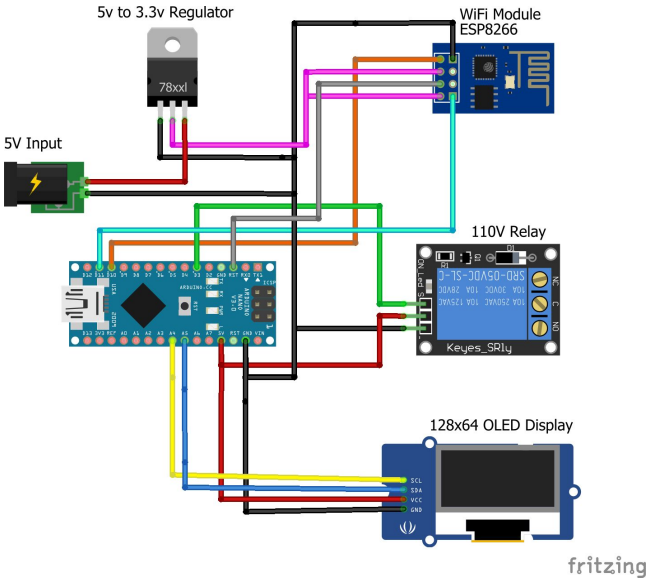



After the above steps were finished components along with the cables were hidden nicely in the black case box.





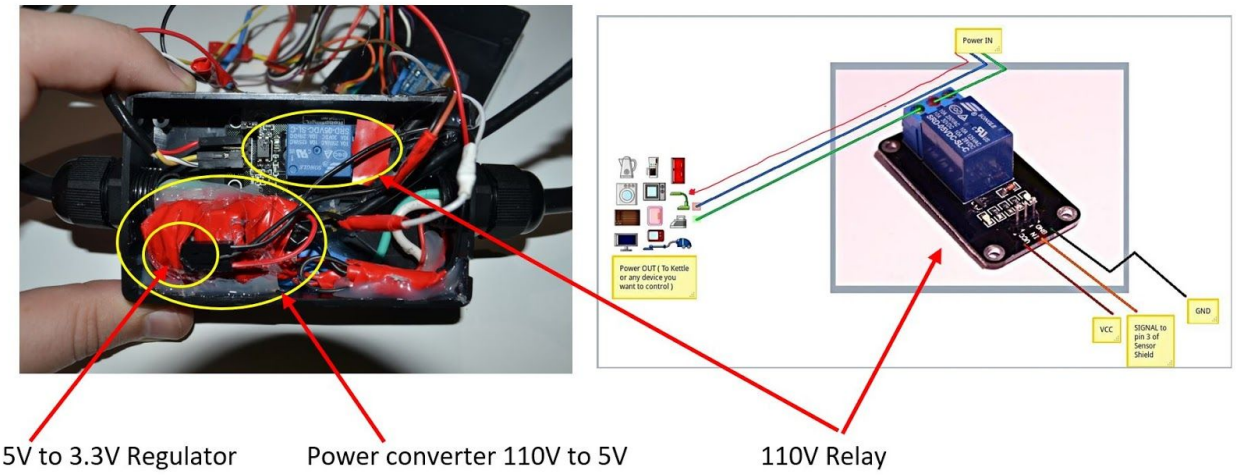
2.3 Power Outlet Unit:



Since we are using Arduino Nano for Power Outlet Unit, we had to make sure that every component for this unit works properly, therefore, we've started building circuit using regular breadboard and Arduino Uno.

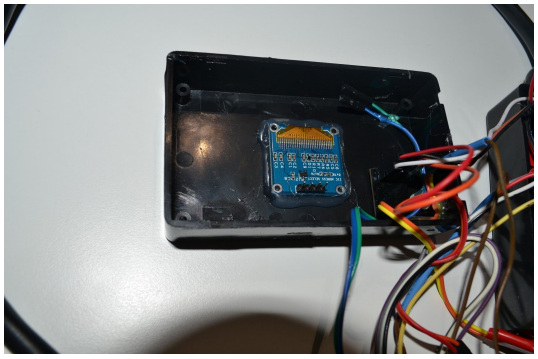
Relay 110V to Arduino:

First step for this part of the project was to cut the extension cord to desired length. Once this was done we have reconnected two wires to "original state" before they were cut. The two ends of the third wire were connected to the relay, see below for the wiring.



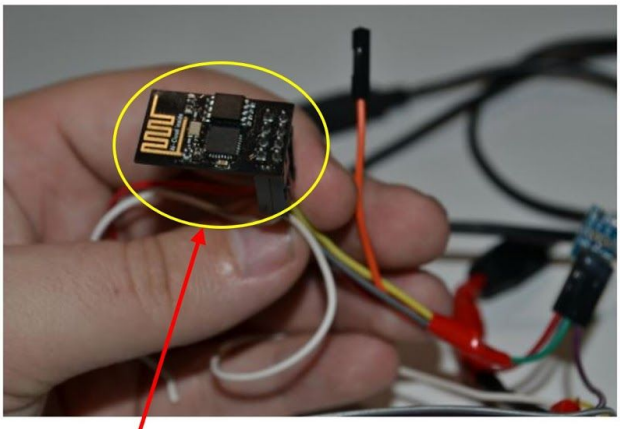
When relay was set, the next step was to connect the Arduino (in this project we used Digital Pin number 4, 5V (VCC) and Ground (GND)). We have written a small program to check if everything is connected and working properly.

128X64 OLED Display:

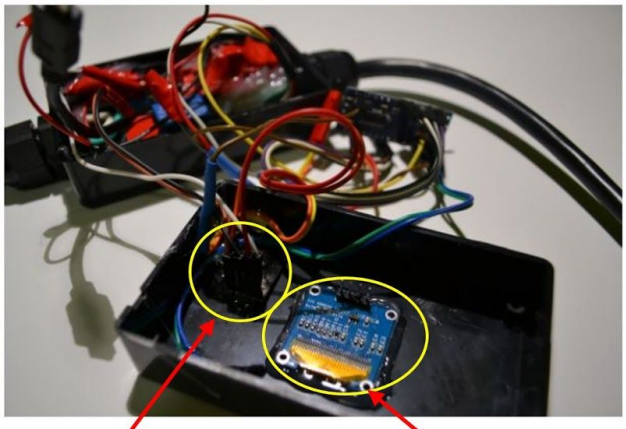


For OLED we used Analog Pins 4, 5, power 5V and GND from Arduino. Connecting Display to microcontroller was not a big challenge, but getting this screen to work, plus finding a library that supports this screen was more challenging. At the beginning we used library from Adafruit, but latter on was changed to U8glib due to conflict with WiFi module. After some trials and errors we were able to print “Hello World” on the screen and with small steps we achieved minimum that was required to move forward. At that point we had fully functional devices – Relay and “Hello World” on the screen.

WiFi Module ESP8266:



WiFi Module ESP 8266



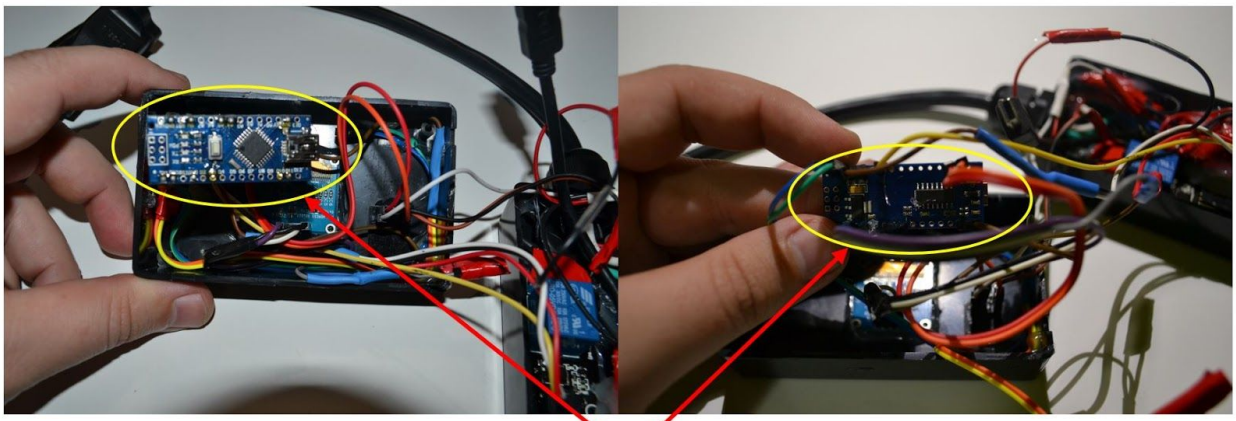
WiFi Module ESP 8266 128X64 OLED Display

According to the schematic provided by the manufacturer, we connected WiFi Module to Arduino. To connect WiFi Module we used Pins D10, D11 and RST. Once this module was connected to the Arduino Uno 3.3V, everything worked in unison. However, when we used Arduino Nano the WiFi we had a problem with voltage. To resolve this situation, we used 5V to 3.3V Regulator that was connected directly to 110V to 5V Power Converter which we took off from a cell phone charger.

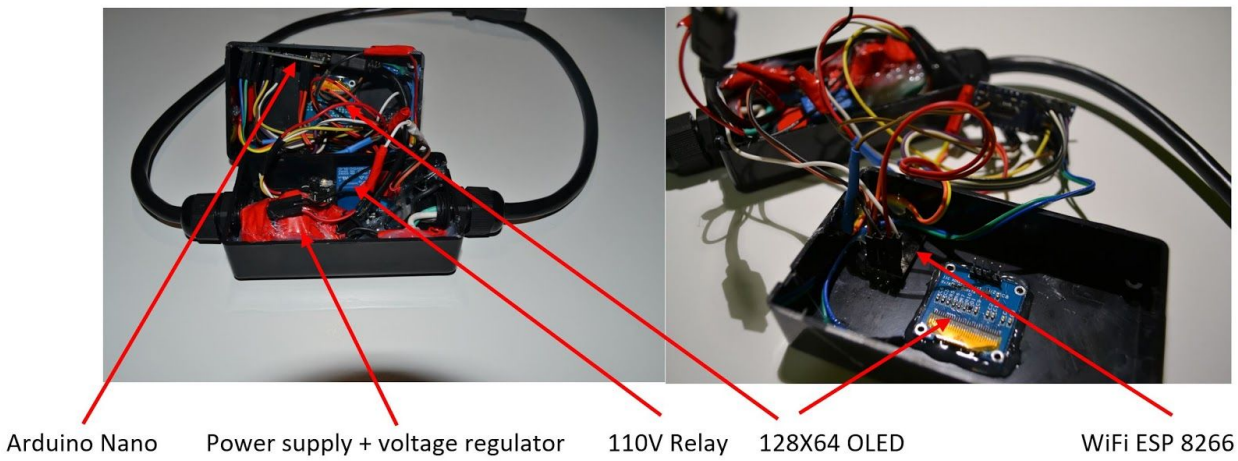
Power Supply:

As a power supply to Arduino and WiFi Module we've used cell phone charger, but we had to do few modifications. We disassembled the charger to obtain power converter 110V to 5V, we soldered few wires to the circuit where the Voltage is already converted from 110V to 5V (due to safety concert this step is not explained*) . After all modifications were completed, we were ready to test it. Figure XX. However, we didn't secure all components on the charger and shield that is around USB cable accidently touched power converter what cause a shortage. Charger number one had to be replaced with charger number two. On the second attempt we succeed and were able to move forward.

Arduino Nano:



Once all components were tested, it was time to switch from Arduino Uno and breadboard to Arduino Nano. We began soldering each device one by one to Arduino Nano performing tests to make sure that everything works. After all devices were joined and functioned as intended, it was time to secure all wires and hide everything in small black box.



3. Lesson Learned:

- Installing different servers one by one is really hard. Most of the servers application were written for x86-x64 machines and later were ported to ARM architecture. Every single server application takes many hours of research and reading how to install and configure it on Raspbian platform. In the future, I would better spend that time to find some bundled package that can install those application with single command, and spend the rest of the time to design and make hardware parts for this project.
- I would buy more capacitors and resistors because some hardware implementations used different parts values/properties that were not included with Arduino Kits.
- Check full device specs before considered it to be included into project. Raspberry Pi can substitute many devices, but in case of our project we needed analog input pins to connect sensors, and it became a problem. If I knew that it does not have analog input I would buy some analog to digital converter to get same 0 to 1023 reading across different devices.
- Use good quality wires for soldering and connection between the devices. In our project we used the same cables that came with Arduino Kit and they sucks. When we soldered one wire and started to solder another, the first wire was holding barely on one tiny wire.
- Multiple devices can communicate using one i2c communication. They can be identified by address.

- Voltage regulator is very helpful thing to use along with external power supply. Especially when there is more components connected to one board and board can have difficulties with supplying the power to all of them.
- Evolutionary development is amazing practice when building system of devices. It is important to follow the following rule: Test, test, and test. Once you are done with testing, test again.
- Keep in mind that some devices can produce heat, which can produce error for other devices readings such as temperature sensor.
- When using libraries verify if capacity of the microcontroller program space can afford library size before you use it. It can save you incredible amount of time in same cases.
- Allow device to be reprogrammed whatever possible. (In example: when using power supply by USB cable)
- Be aware of explosions.
- Never connect voltage regulator 5V to 3.3V directly to 110V
- Working in the group can benefits teammates since each member has a different knowledge and can pass his wisdom to others.

4. Resources Used:

Raspbian OS download:

https://www.raspberrypi.org/downloads/raspbian/

Software to make micro-SD card bootable with Raspbian OS:

https://rufus.akeo.ie

DNSMASQ server application installation and configuration help:

https://www.the-hawkes.de/dnsmasq-a-local-dnsdhcp-server-on-raspberry-pi.html

HTML5 template for Main Unit interface:

https://html5up.net/hyperspace

Convert analog signal to digital. Used to connect light resistor to Raspberry Pi.

https://www.allaboutcircuits.com/projects/building-raspberry-pi-controllers-part-5-reading-analog-data-with-an-rpi/

DHT11 sensor readings using Python code:

https://github.com/szazo/DHT11_Python

BMP180 Barometric Sensor: Wiring and general idea how it works.

http://randomnerdtutorials.com/guide-for-bmp180-barometric-sensor-with-arduino

Multiple devices on one i2c bus: How to handle 2 devices via one i2c bus.

https://www.youtube.com/watch?v=QQLfzlPGjjE

DHT11 Digital Humidity Temperature Sensor: Wiring and general idea how it works.

http://www.circuitbasics.com/how-to-set-up-the-dht11-humidity-sensor-on-an-arduino