MINISTRY OF EDUCATION

TECHNICAL UNIVERSITY
OF CLUJ-NAPOCA, ROMANIA

FACULTY OF AUTOMATION AND COMPUTER SCIENCE

AUTOMATIC GREENHOUSE ROOF

SEMESTER PROJECT

Authors: **Bădău Elena Nicoleta**
Zah Elena

Scientific coordinator: **Sita Ioan Valentin**

2022

Table of contents

1 Introduction

1.1 Project context

The main focus of our project is to make use of our accumulated knowledge during our university educational training and incorporate it in a practical, easy to acces and to further develop IoT automated system.

The automatic greenhouse roof aims to describe a useful, yet technologically improved task for any building used in agriculture, touristic greenhouses or even in our very own homes. Furthermore, it illustrates the never-ending usage and accesibility of IoT both in educational and industrial purposes.

1.2 Objectives

Our target is to implement a demo greenhouse roof system that can be remotely supervised and controlled using IoT electrical and software components.

We will try to simulate a control of the state of the window shades in relation to the light intensity needed inside the greenhouse. Consequently, if the light is too strong for the plants, the motors should be actuated to roll or cover the windows and if the lights lower under a specified threshold, the motors should roll out or remove the shades from the windows.

1.3 Specifications

The main component of the project is ESP8266 WiFi Module is a self contained SOC with integrated TCP/IP protocol stack that can give any microcontroller access to your WiFi network. The ESP8266 module is an extremely cost effective board with a huge, and ever growing, community.

2 Implementation

2.1 Hardware configuration

The circuit diagram which will be implemented using physical components was implemented using Tinkercad and can be seen in Fig. 2.1.1.

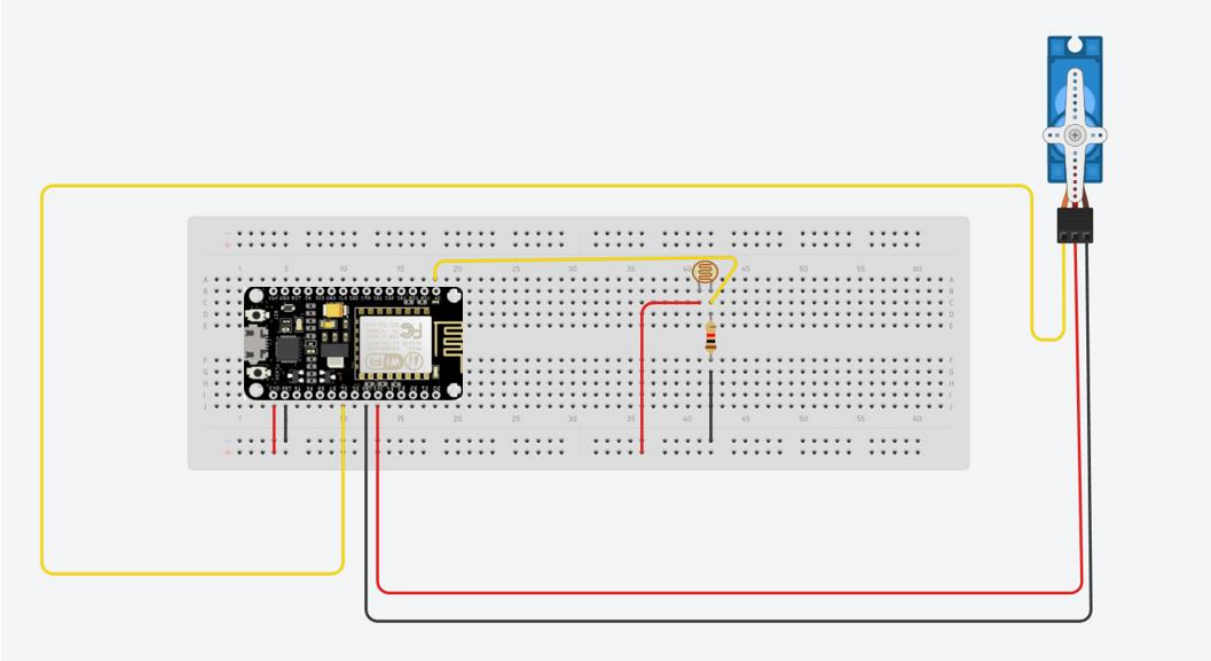


Fig. 2.1.1. Access Control System – circuit and wiring diagram

The components used in the hardware configuration can be seen in Fig. 2.1.2 – 2.1.6:

Fig. 2.1.2. ESP8266 WiFi Module

Fig. 2.1.3. Mini servo motor

Fig. 2.1.4. Photoresistor Sensor

Fig. 2.1.5. Resistor

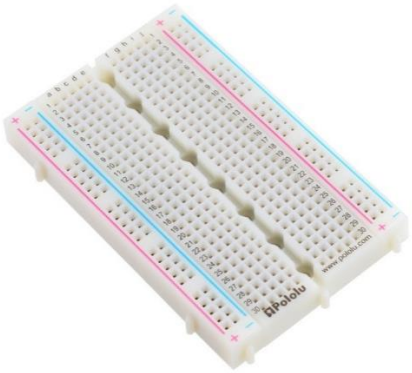


Fig. 2.1.6. Breadboard

We mounted all these components in a circuit, it can seen in the Fig. 2.1.7.

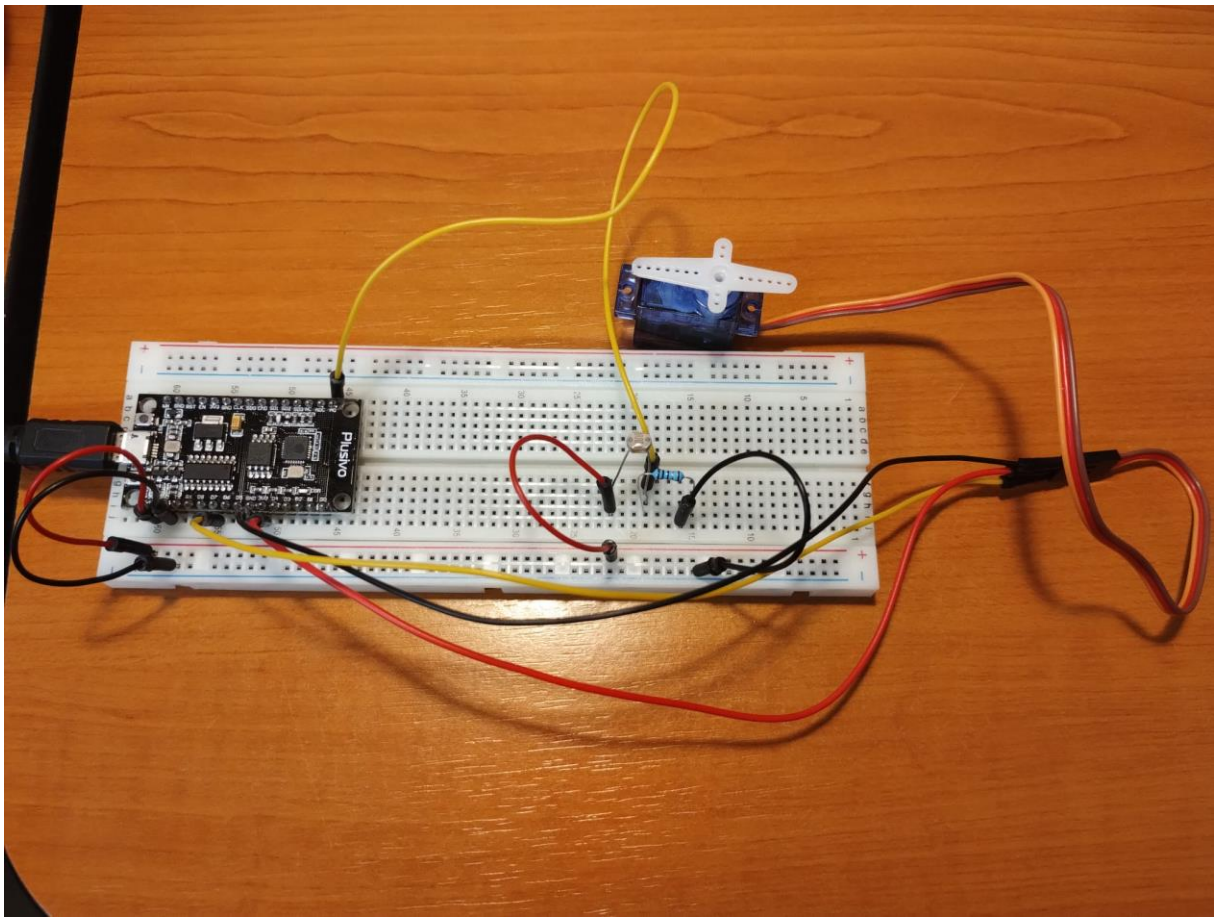


Fig. 2.1.7. Our Configuration

2.2 Arduino + Web Server

The development environment used for for writing the code in this project is Arduino IDE.

The following part is focused on the main program (for the central Arduino board).

The libraries used by this application are:

```
#include <Servo.h>;
#include <ESP8266WiFi.h>
#include <ESP8266HTTPClient.h>
#include <WiFiClient.h>
#include <HttpClient.h>
```

The first elements to be declared and initialized were constants of type char like ssid used for the network's name (Service Set IDentifer), password used for the password of network, host for Ipv4 address, also constants of type int like port for Domoticz port and watchdog used for frequency of sending data to Domoticz.

```
const char* ssid     = "Elena's iPhone";
const char* password = "        ";
const char* host = "172.20.10.2"; // IPv4 adress
const int   port = 8080;     //port Domoticz
const int   watchdog = 6000; // Frequency of sending data to Domoticz
unsigned long previousMillis = millis();
```

Also, we declared 3 variables that uses libraries, http for HTTPClient, wifiClient for WiFiClient and motorPin for Servo. The variable of type String, url, is used for url address. The photoresistor uses as input an analog pin A0, for that we need an int type variable that takes the analog data from the sensor.

```
HTTPClient http;
WiFiClient wifiClient;
String url;

Servo motorPin;
int sensor_pin_light = A0; // Photoresistor input at Analog PIN A0
int output_value_light = 0;
unsigned long current_ms=0;
```

Next up, we created a function to retrieve data from the light intensity sensor, reading data from it is analog:

```
void read_sensor()
{

  Serial.println("Reading From the Sensor ...");
  output_value_light= analogRead(sensor_pin_light);
 //output_value_light = (output_value_light/1024);
 Serial.println("Light : ");
 Serial.println(output_value_light);
 delay(3000);
}
```

When the light intensity value is lower than 20, the servo motor will move at an angle of 90 degrees. For this we created a function:

```
void roof()
{
  if (output_value_light<20)
  {
  motorPin.write(90);
  delay(1000);
  motorPin.write(0);
  delay(1000);
  }
}
```

To transmit data through the WiFi module, ESP8266, we created a function in which the connection is made with the IP address of the network and the data will be transmitted through a url address to the Domoticz platform:

```
void sendDomoticz(String url) //updating data
{
  Serial.print("connecting to ");
  Serial.println(host);
  Serial.print("Requesting URL: ");
  Serial.println(url);
  http.begin(wifiClient,host,port,url);
  int httpCode = http.GET();
    if (httpCode)
    { if(httpCode == 200)
     {
        String payload = http.getString();
        Serial.println("Domoticz response ");
        Serial.println(payload);

    }
    }
  else
    //Serial.println("Not connected to Domoticz");
    Serial.println (httpCode);
   // Serial.println("Nope, closing connection");

  http.end();
}
```

Next, the setup:

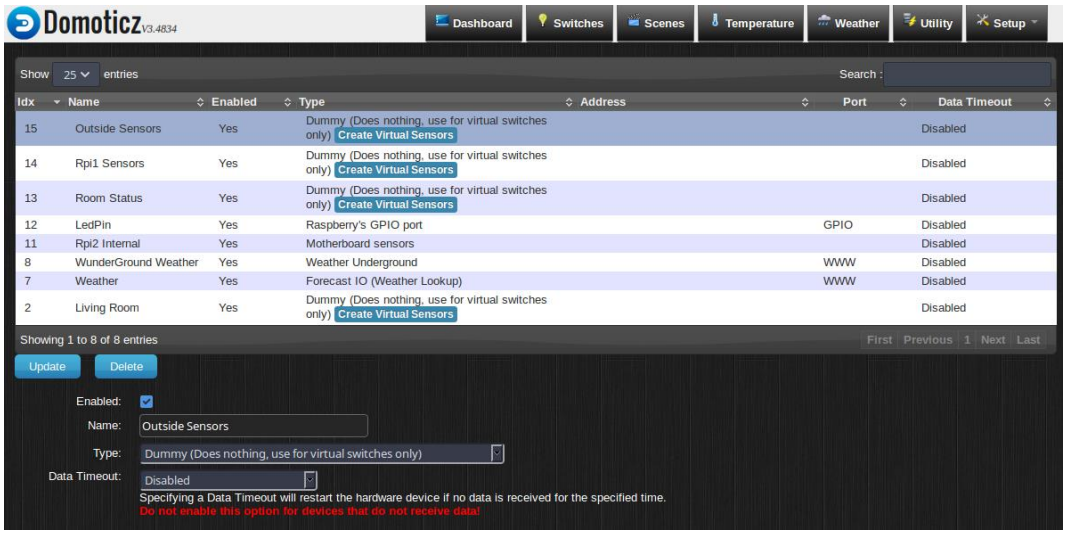
```
void setup()          // put your setup code here, to run once:
{

  Serial.begin(9600);
  Serial.println("The board has started");
  motorPin.attach(D7);
  delay(2000);
  current_ms=millis();

  //Connecting to Wifi

   Serial.println("Connecting Wifi...");

    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED)    //connecting to wifi
      {
        delay(500);
        Serial.print(".");
        Serial.println("");
        Serial.println("WiFi connected");
        Serial.println("IP address: ");
        Serial.print(WiFi.localIP());
      }

      delay(10);
}
```

Also, the Loop:

```
void loop()
{
  read_sensor();
  roof();
  unsigned long currentMillis = millis();
  if ( currentMillis - previousMillis > watchdog )
  {
    previousMillis = currentMillis;
    if(WiFi.status() != WL_CONNECTED)
      {
        Serial.println("WiFi not connected !");
      }
    else
      { Serial.println("Send data to Domoticz");
        //int sensorValue = analogRead(A0);
        // read the input on analog pin 0
        //int output = sensorValue * (100 / 1023.0);
        // Convert the analog reading (which goes from 0 - 1023) to a voltage (0 - 100V)
        //request command
        url = "/json.htm?type=command&param=udevice&idx=1&nvalue=0&svalue=";
        url += String(output_value_light); url += ";";
        sendDomoticz(url);
       }
    }
}
```

3 User's manual

Fig. 3.1.1. Domoticz platform



In your Domoticz server, access Hardware in the Setup menu. Now we see a list of current created devices in Fig. 3.1.1 . The sensor will not be directly connected to Domoticz, it will be a remote sensor sending data to a Virtual device on the server for that purpose. On this page name a new Hardware. This can have any name being a dummy device type. After this was added, the new hardware will show on the list and appear the "Create virtual sensors" button. Click that and select the type of device you want. For this project we have used a light sensor type . After clicking OK, go to Setup -> Devices. The new sensor should appear on your devices list.

Pay attention to the IDX number. To this the device index we want to send data from the ESP8266 to the Domoticz Server. The IDX number should be parsed to your JSON string in your SendDomoticz() function. You can see our result obtained in the course of few hours in the figure below Fig. 3.1.2.

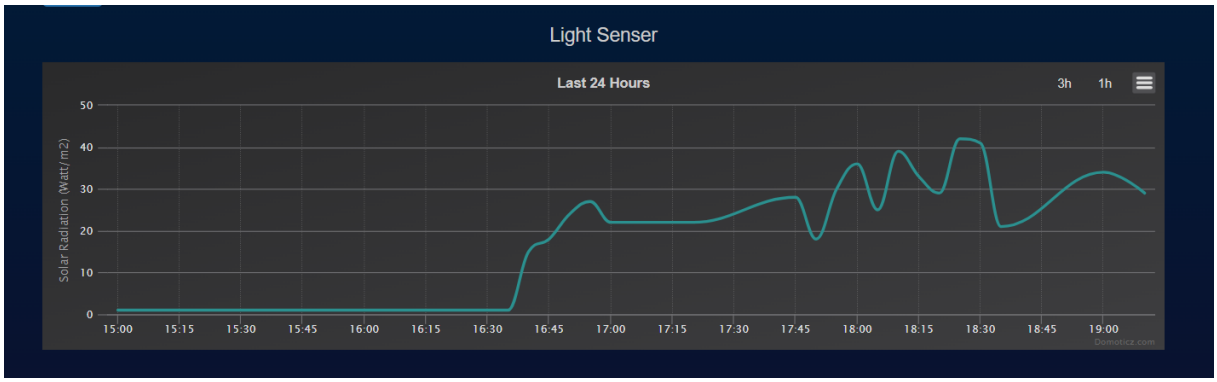


Fig. 3.1.2. The resulting graph

4 Conclusions

4.1 Results

The actuating system reacts automatically to its intended purpose and completes the task as programmed.

4.2 Possibilities of improvement/further development

Some possibilities of further improvement of this project might include:

- Include other sensors and actuators (temperature, soil or air humidity) and corresponding actuators
- Verify with additional sensors if the actuators behaved as programmed (position sensor)
- Alarm system if the greenhouse is over or underheated, extend with a set of notification implementation.

5 Bibliography

[1] J. Enokela, T. Othoigbe, „An Automated Greenhouse Control System Using Arduino Prototyping Platform". Available:

https://facades.lbl.gov/cec-electrochromics/refs/LBNL_electrochromics_14.pdf

[2] J. Carmody, S. Selkowitz, E. Lee, D. Arasteh, T. Willmert, „Window System for High-Performance Buildings". Available:

https://facades.lbl.gov/cec-electrochromics/refs/LBNL_electrochromics_14.pdf

[3] T. Saha, M.K.H. Jewel, M.N. Mostakim, N.H. Bhuiyan, „Construction and Development of an Automated Greenhouse System Using Arduino Un", May 2017, [Online]. Available:

Construction_and_Development_of_an_Autom.pdf

[4]https://www.youtube.com/watch?v=dcKo_XnSvEU&ab_channel=TinkeringLabsOnline

[5] https://www.youtube.com/watch?v=pm7CAt1kRnA&ab_channel=Prashavlogs

[6] https://www.youtube.com/watch?v=r8JdL1HYhy0&ab_channel=AchmadKhaerudin