PIC - ElectroCap

Electrical and Computer Engineering

Software Development - Report

Authors:

Lucas Nunes dos Santos (102659) lucas.dos.santos@tecnico.ulisboa.pt
João Rodrigues Ribeiro (102715) joao.r.ribeiro@tecnico.ulisboa.pt

Group 13

2023/2024 – 2nd Semester, P4

1 Objective

There has been an effort from part of our team (João and Lucas) to develop a working iteration of the needed software to perform the various tasks that our product promises to execute. Our software is divided into two parts: the first one is the code for the Arduino, that focuses on the rotation of the blinds and the solar tracking and the second part is the code for the ESP8266 that allows Wi-Fi communication needed for the web app.

2 Arduino Software

The software implemented for the arduino UNO is responsible for the movement of the blind and the calculation of the current solar altitude, used for the automatic mode.

The program starts by receiving the current date and time, sent by the ESP8266, as shown below.

```
String Time = mySerial.readString();
```

We then separate this time string into different variables containing the time and date separately. This is done so we can be able to set the time in the machine, and never have to go through this process again, since the internal clock of the Arduino keeps track of the time after being set.

We then come to the main loop of the program, that will be constantly ran, and will deal with the automatic and manual modes of the product.

It starts by checking if it has received any message from the ESP8266, since this means that the user has changed the mode of functioning, or the inclination of the blinds.

```
if(mySerial.available()!= 0){
   mode_rec = mySerial.readStringUntil('\n');
   mode_rec = mode_rec.substring(1,2);
   elevation_rec = mySerial.readStringUntil('\n');
   elevation_rec = elevation_rec.substring(2,4);
}
```

We then check which mode is currently selected, and execute said mode, by calling its respective function.

```
if(mode_rec=="1"){
   automatic_mode();
}
else{
   rotate_blinds(elevation);
}
```

The automatic mode starts by calculating the current inclination of the sun. This is done with the use of a library called SolarCalculator. This library has a function that when provided the current date, time and spacial coordinates of the panel, returns the elevation of the sun in degrees. It also returns the azimuth, but we will not be using it in this project.

```
void automatic_mode(){
   time_t utc = now(); //Get the time and date of the measurement
   double azimuth = 0, elevation = 0;

   calcHorizontalCoordinates(utc, latitude, longitude, azimuth, elevation);
   Serial.print("Elevation = ");
   Serial.println(elevation);
   rotate_blinds(elevation);
}
```

In the end of this function we call upon another function responsible for rotating the blinds, that is used in both modes. This function receives an elevation and is responsible for effectively actuating on the step-motor responsible for the rotation of the panels.

This function starts by calculating the variation of inclination of the panel. This will be called γ (gamma) and is given by the expression bellow.

$$\gamma = elevation - current\_angle \tag{1}$$

Gamma represents how much, and in which direction, must the panel rotate, to meet the desired elevation.

We then check if the rotation is smaller then 0.71, since this value isn't high enough to produce any real rotation on the step-motor.

```
if(abs(gamma) < 0.71){
   return;
}
```

If the value of gamma is bigger than 0, we will need to rotate upwards, since the desired elevation is bigger than the current one. We also need to check if the gamma is bigger than 60°, since the motor cannot actuate on values over that. If that is the case, we divide gamma by 2 and rotate the panel twice.

```
gamma = elevation - current_angle;

if(abs(gamma) < 0.71){
   return;
 }
 current_angle = elevation;

 if(gamma > 0){ //The angle of the panel is getting smaller by minus gamma
    gamma_aux = gamma;

    if(gamma > 60){
       gamma = gamma/2;
    }
```

At last, we will call upon the function that rotates the amount of notches that correspond to the desired angle, by performing sequential digital writes to the motor. The number of notches rotated is calculated using the following expression.

$$rotating\_notches = \left\lceil \frac{\gamma \cdot 512}{360} \right\rceil \tag{2}$$

If the angle gamma is negative we repeat the process, but the function used to rotate the blinds inverts the order of rotation, so it rotates downwards.

For the manual mode, we repeat the steps refereed previously, but we skip the calculation of the solar elevation, since the variable elevation is already provided by the user.

```
if(mode_rec=="1"){
   automatic_mode();
}
else{
   rotate_blinds(elevation);
}
```

In order to calculate the power production, it was introduced a resistor of about 39 Ω at the end of the circuit. The voltage that passes through that resistor is read by the Arduino, with a small conversion, needed to obtain the value in volts:

```
voltage_in  = (value_in * ref_voltage) / 1024.0;
voltage_out  = (value_out * ref_voltage) / 1024.0;
```

As the value of the resistor is known, the current is obtained by applying Ohm's Law and, therefore, it is possible to calculate the power obtained by multiplying voltage with current.

```
in_current = (voltage_in - voltage_out) / R;
in_power = voltage_in * in_current;
```

After the calculations, the data is sent from the Arduino to the ESP8266.

3 ESP8266 Software

The software for the ESP8266 was developed in the *Arduino Cloud*, so that the ESP is capable of receiving the inputs from the user via the dashboard in a wireless way.

Before going on about the workings of the software it is important to present the dashboard of the application developed.

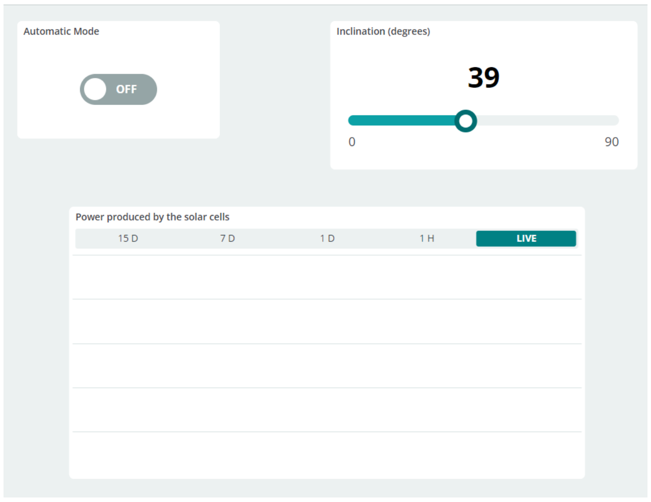

Figure 1: Arduino Cloud dashboard

The solar blinds have two operation modes. In the automatic mode the elevation of the blinds is automatically changed in order to maximize the energy produced. However, there is also a manual mode that allows the user to adjust the blinds, selecting the desired elevation angle. The switch presented in the dashboard defines the activated mode: when it is ON, the solar blinds are in automatic mode and when it is OFF they are in manual mode. The inclination is then adjusted by the user in the slider, transmited to the Arduino and, consequently, the motor, if the solar blinds are in manual mode.

Lastly, there is a graph that allows to see the power produced by the solar cells as a function of time. As it will be analysed later on, this data is sent from the Arduino to the ESP.

As previously referred, the ESP starts by sending the date and time (in UTC format) to the Arduino. This allows the solar elevation to be calculated correctly from the SolarCalculator library.

```
void loop() {
  ArduinoCloud.update();
  if (a == 0) {
    write_time();
    a = 1;
  }

  delay(1000);
}
```

The function `write_time()` writes the time and date to the Arduino in UTC format, through the `getEpochTime()` function. `a` is a global variable defined at the beginning of the program as `a = 0`, to ensure that the time and date is only written in the first iteration of the software.

The variables in Arduino Cloud were created to be read and write variables, which means they can be sent from the ESP to the Arduino and vice-versa, and were established to have an update every time there is a change in the value attributed. To have the operation described above, every time the automatic mode switch is changed, both the inclination value and the selected mode need to be sent to the Arduino. That is done through the following function:

```
void onAutomaticModeChange()  {
  mode_send = String(automatic_mode);
  mode_send.toCharArray(ptr1, sizeof(mode_send));
  Serial.write(ptr1);
  Serial.println(mode_send);
  inclination_send = String(inclination);
  inclination_send.toCharArray(ptr2, sizeof(inclination_send));
  Serial.write(ptr2);
  Serial.println(inclination_send);
}
```

Since the read function is prepared to receive strings, it is needed to change the boolean value of the state of the switch to a string, which will then be sent to the Arduino. The same processing is done to change from a float to a string, the inclination. In order to facilitate the reception in the Arduino of the strings received, and in what conditions, if only the inclination value is altered, both the inclination and the state of the switch are sent to the Arduino, such as `onAutomaticModeChange()`and `onInclinationChange()` are identical.

The Arduino calculates the power as previously referred and sends it to the ESP, which receives the string and then displays it in the graph in the dashboard.

You can check all our software in our GitHub repository: SolarBlinds.