Internet of Things-Based Solar Tracker System

Aboubakr El Hammoumi, Saad Motahhir, Abdelaziz El Ghzizal, and Aziz Derouich

Abstract Internet of Things (IoT) technologies, along with economies of scale and advances in hardware, software, and network technologies, have accelerated the explosion of connected objects across the Internet. A connected object can be controlled online from an IoT platform and can send, receive, and process various and varied data. In this chapter, we leverage some of the IoT technologies to propose a simple and low-cost IoT solution to monitor and control a smart dual-axis solar tracker system for performance evaluation. The solution also includes alert notifications to inform a remote user through phone or mail (or both) when a sensor has reached a certain predefined event. The solution is designed based on low-cost and easy-to-use hardware and software and an online open-source IoT platform. The design aspects of the IoT-based solar tracker are extensively described in this chapter. Moreover, a prototype of the IoT-based solar tracker has been manufactured and tested. Test results demonstrate that solar tracker data can be sent easily and properly and can be directly monitored online, as well as the solar tracker, can take commands from the IoT monitoring application.

Keywords Internet of things · IoT · Monitoring · Arduino · Solar tracker

A. El Hammoumi (✉) · A. El Ghzizal
InnovativeTechnologies Laboratory, EST, SMBA University, 30000 Fez, Morocco
e-mail: aboubakr.elhammoumi@usmba.ac.ma

A. El Ghzizal
e-mail: abdelaziz.elghzizal@usmba.ac.ma

S. Motahhir
Engineering, Systems, and ApplicationsLaboratory, ENSA, SMBA University, 30080 Fez, Morocco
e-mail: saad.motahhir@usmba.ac.ma

A. Derouich
Industrial Technologies and Services Laboratory, EST, SMBA University, 30000 Fez, Morocco
e-mail: aziz.derouich@usmba.ac.ma

S. Motahhir and A. M. Eltamaly (eds.), *Advanced Technologies for Solar Photovoltaics Energy Systems*, Green Energy and Technology,
https://doi.org/10.1007/978-3-030-64565-6_4

1 Introduction

The International Telecommunication Union (ITU) has defined the IoT as a global infrastructure for the information society that enables the provision of advanced services by connecting (physical and virtual) things, based on existing and evolving interoperable information and communication technologies [1]. The IoT, or as it is called the Internet of Everything (IoE), includes all devices that can communicate with the Internet and that can collect, send and process the data they capture from their surrounding environment using embedded sensors and processors in addition to the communication networks [2, 3]. IoT applications are expected to equip billions of objects with connectivity and intelligence [4]. It is already being deployed extensively, in various fields, namely: wearables [5], smart buildings [6], smart cities applications [7], health care [8], agriculture [9], industrial automation [10], solar monitoring systems [11], etc. In this chapter, we leverage some of the IoT technologies to design and build an IoT-based solar tracker system, where an IoT application is proposed to control and monitor this system.

To maximize the absorption of sunlight and thereby increasing energy production, it is necessary to integrate solar tracker systems into conventional solar energy systems, where the solar panels can be fixed on a structure that moves according to the sun's path. 10–50% additional output energy can be obtained by using solar tracker systems that track the sunlight instead of conventional systems that attach at a fixed angle [12]. Depending on the mechanisms used to orient the solar panels, solar tracker devices can be divided into single or double axis devices. Single-axis devices can only track sunlight by rotating around a horizontal or vertical axis, i.e. they track the sun's movement in one direction (toward East and West or toward South and North). While dual-axis solar tracker devices can rotate vertically and horizontally to ensure solar panels are always perpendicular to the sunlight [13]. Various solar tracker systems have been reported in the literature and they differ according to employed tracking methods [14]. For instance, a sensor-based solar tracker has been proposed in our previous works, it uses light sensors to predicts the sun's position (intensity of light) to track the sun for maximum power generation [15, 16]. The solar tracker system detects the sun position with the help of Light Dependent Resistor (LDR) sensors and sends the data to the controller. This latter then processes these data to command two servomotors that rotate a photovoltaic (PV) panel, in the optimal directions, to move toward the sunlight. For more details, authors in [17] have categorized solar tracker systems based on five tracking methods: sensor-based tracker method, geometric and astronomical equation-based method, open- or closed loop-based method, artificial intelligent-based method, and a combination of two or more of these methods. Indisputable that solar tracker systems have manifested a high ability for increasing the efficiency of solar panels to produce more energy. Besides, making a solar tracker device as a connected object using IoT technologies can be more profitable and advantageous, where the user can remotely control the device and access its data, including the electrical and environmental parameters linked to the solar panels, from an IoT platform. These data can be used to evaluate the solar tracker operation, as well as to assess the PV energy potential, early detection and diagnosis of electrical faults, evaluate the weather variations, and preventive maintenance.

However, to the authors' knowledge, there are only a few attempts available in the literature that deal with this subject. Authors in [18] have developed a dual-axis solar tracker with IoT monitoring using the Ubidots IoT platform. A WiFi ESP8266 board has been employed to connect their tracker device with the internet to communicate with the IoT monitoring application, where its data, including voltage, current, and power are displayed. The same WiFi board has been used by the Authors in [19], who have elaborated a single axis solar tracker, to send the same mentioned data to the cloud server of the ThingSpeak IoT platform so that they can be visualized in a dashboard that preconfigured on ThingSpeak. Furthermore, a monitoring solution of a solar tracker using Raspberry Pi3 (RPi3) board and a personal developed cloud server has been established in [20]. It uses socket programming using Python language to communicate between the "client" that runs on a remote laptop and "server" that runs on RPi3. There are two ways to design an IoT monitoring platform, either we design it ourselves, or using one of the available IoT platforms, which most of them are open source. By using the second way, the development process of an IoT project can be done easily and as early as possible. Because IoT platforms are designed to reduce an IoT project development time by enabling ready-made, reusable technology stack and are compatible with and support various hardware platforms (such as Arduino *and* Raspberry) [21]. To this end, IoT platforms are widely used by engineers and researchers in their IoT projects [22–25].

This chapter aims to present a simple and low-cost IoT solution to monitor and control a dual-axis solar tracker system. A low-cost and popular embedded board (Arduino) is used along with LDR sensors, servomotors, and associated circuits to control a PV panel to track the sunlight for maximum power generation. Different sensors are employed to measure electrical output parameters (voltage, current, and power) and environmental parameters (temperature, humidity) linked to the solar tracker system. An Ethernet shield is used to connect the system over the Internet and to exchange data between hardware and the cloud server using Message Queuing Telemetry Transport (MQTT) protocol. Data processing and activities that occur can be monitored online through an IoT monitoring application developed on Cayenne IoT platform. The solar tracker can also take commands from the monitoring platform. In addition, the application includes an alert system to notify the user when a sensor has reached a certain predefined event. The hardware and software used have been chosen to be simple and inexpensive. Arduino board is used due to its low-cost and its easy-to-use hardware and software [26, 27]. Likewise, Cayenne IoT platform is used due to its easy-to-use interface and protocols. It is an open-source IoT platform that has a simple Application Programming Interface (API) to store and retrieve data from things using the MQQT protocol over the Internet or via a Local Area Network [28].

The rest of this chapter is structured around three sections. Section "Research methodology" describes the architecture of the proposed IoT-based solar tracker system and presents the hardware and software used to develop it. Section “Results and discussion” lists and discusses the experimental results. Finally, the main conclusions of this chapter are drawn in Section “Conclusion”.

2 Research Methodology

2.1 System Description

The proposed IoT-based solar tracker system is depicted in Fig. 1. It is a dual-axis solar tracker that can rotate automatically to track the sun position using LDR sensors, or manually by the user through the dashboard of an IoT application. The system starts with detects the sun position (intensity of light) by LDR sensors and sends the data to the controller (Arduino Mega board). This latter then processes these data to command servomotors (SM1 and SM2) that hold the PV panel to rotate toward the sun. The values of the generated PV voltage and current, temperature, and humidity are also sent to the Arduino through associated sensors. Next, the Ethernet shield, which is mounted with Arduino and allows it to be connected to the Internet, will send the data that has been taking and/or processed by Arduino to the cloud (webserver). Lastly, the solar tracker data, including LDR sensors, PV power, temperature, and humidity, are displayed in real time in the IoT monitoring application via pre-created Widgets. The IoT monitoring application is designed using Cayenne myDevices platform. Once the user is connected to the internet from his computer or smartphone, he can visualize, in the dashboard of the IoT application, all solar tracker data in their associated widgets. Therefore, the user has the necessary data linked to the environment and performance of the PV panel. In addition, in the manual mode, the servomotors will take angle directions from their associated widgets in the dashboard. Hence, the user can control his system to seek the best

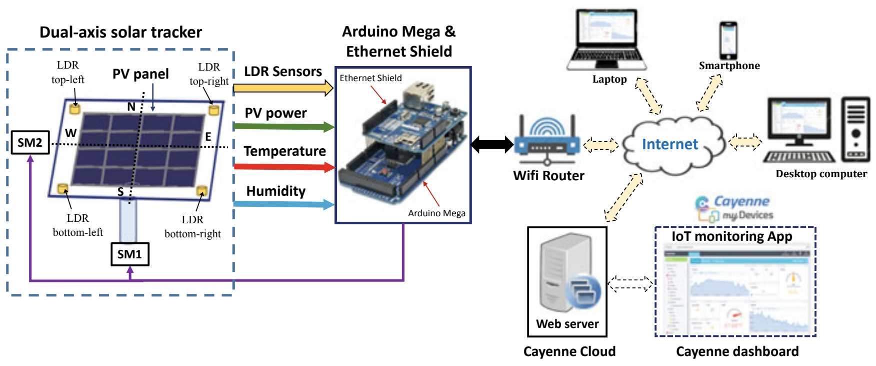


Fig. 1 Schematic of the IoT-based solar tracker system

environmental conditions and extract the maximum energy from the PV panel. The IoT application is also programmed to send notification alerts (SMS or Email) when a senor reaches a predefined threshold value.

2.2 Hardware Design

As shown in Fig. 2, the IoT solar tracker system consists of the PV panel, two servomotors, four LDR sensors, a voltage divider circuit, temperature and humidity sensor, a Led and the Arduino Mega board.

The used PV panel is 115 by 85 mm in size with a 1.6 W output and can generate a voltage up to 6 V [29]. Two 180° servomotors are used to motorize the solar tracker and they are controlled by the Arduino board through PWM pins 5 and 6. The left-right (L-R) servomotor (MG996R) rotates the solar tracker on the vertical axis (East/West), while the Up-down (U-D) servomotor (SG90) rotates the solar tracker on the horizontal axis (South/North).

Four LDRs (Cds GL5528) are used to sense the sun's position and which have been fixed in the four corners of the PV panel. The LDR sensors are connected to the Arduino through analog pins from A0 to A3. The LDR is a resistor whose value decreases with increasing light intensity incident on its surface. The LDR sensor is designed as a voltage divider circuit as can be seen in Fig. 2. The output of the voltage divider is connected to an analog input (A0 for instance) of the Arduino. Then, the

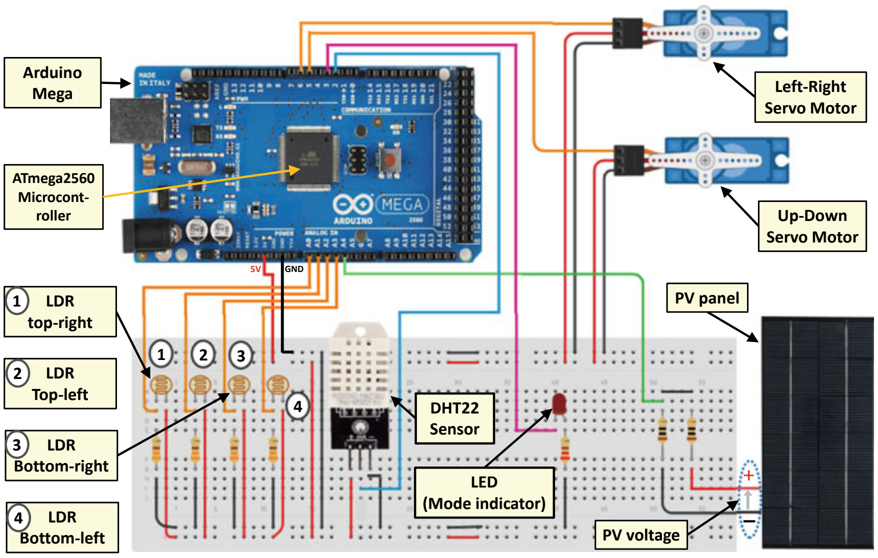


Fig. 2 Electronic circuit of IoT-based solar tracker system

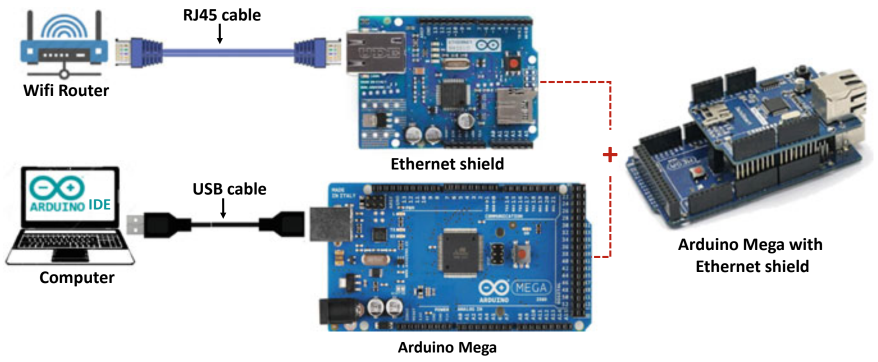


Fig. 3 Hardware interface between Arduino and Ethernet shield

Analog to Digital Converter (ADC) of the microcontroller converts the analog value read by A0 into a digital value between 0 and 1023 because the ADC is coded in 10 bits. The value of the series resistor in the LDR sensor circuit is 330 Ω.

The temperature and humidity are measured through the DHT22 sensor, which is an ultra-low-cost sensor that is widely used in embedded projects. DHT22 has a thermistor and a capacitive humidity sensor embedded in it to measure temperature and relative humidity. Its temperature range is from −40 to 80 °C with $< \pm$ 0.5 °C of accuracy, and its humidity range is from 0 to 100% with ± 2% (Max ± 5%) of accuracy [30]. This sensor uses one signal wire to transmit data to Arduino (digital pin 2), and two wires for power supply.

The PV voltage and current are measured through a voltage divider circuit that acts also as a load and which consists of two series resistors of 10 Ohms. The divider circuit output is connected to the Arduino's analog pin A4. Furthermore, a LED, which is connected to digital pin 3, reflects in the system circuit the mode state of solar tracker (manual or automatic).

The Arduino Mega with ATmega2560 microcontroller is used as the embedded controller that interacts with the Arduino Ethernet shield along with the monitoring platform. The Ethernet shield, which is mounted above the Arduino board, must be connected with a Wi-Fi router (or PC) through an RJ45 cable as shown in Fig. 3. The Ethernet Shield is based on the Wiznet W5100 Ethernet chip that provides a network (IP) stack for TCP and UDP protocols [31].

2.3 Prototype

Figure 4 presents the solar tracker prototype in its detached and assembled state. It consists of the PV panel, the L-R, and U-D servomotors and LDR sensors. The panel is attached to the U-D servomotor on one side and with a bearing on the other side to ensure better flexibility when the solar tracker rotates around the horizontal axis. The

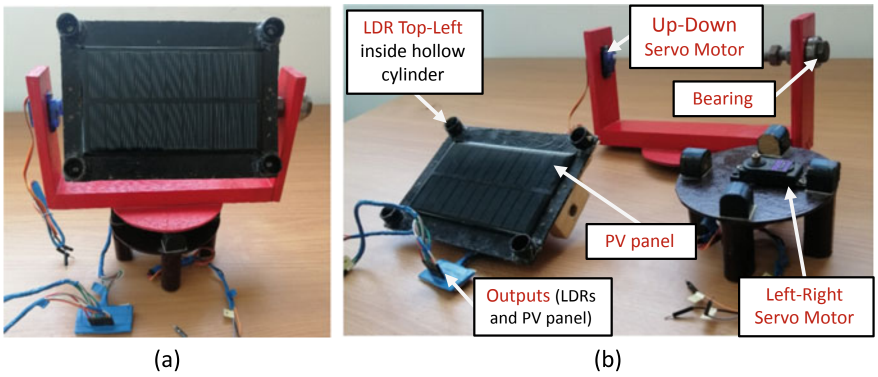

LDR Top-Left inside hollow cylinder
Up-Down Servo Motor
Bearing
PV panel
Outputs (LDRs and PV panel)
Left-Right Servo Motor
(a)
(b)

Fig. 4 Solar tracker prototype in its detached and assembled state

assembly is attached to the L-R servomotor. The LDR sensors are fixed in the four corners of the panel inside hollow cylinders. If the panel is not perpendicular to the sun, at least one LDR will be covered by shadow caused by the surrounding cylinder. Hence, there will be a difference in light intensity. The best orientation is when the light intensities are equal in all LDR sensors. Figure 5 shows the entire prototype of the IoT-based solar tracker system, and it is clear that all reported components in the hardware part have been used to build it.

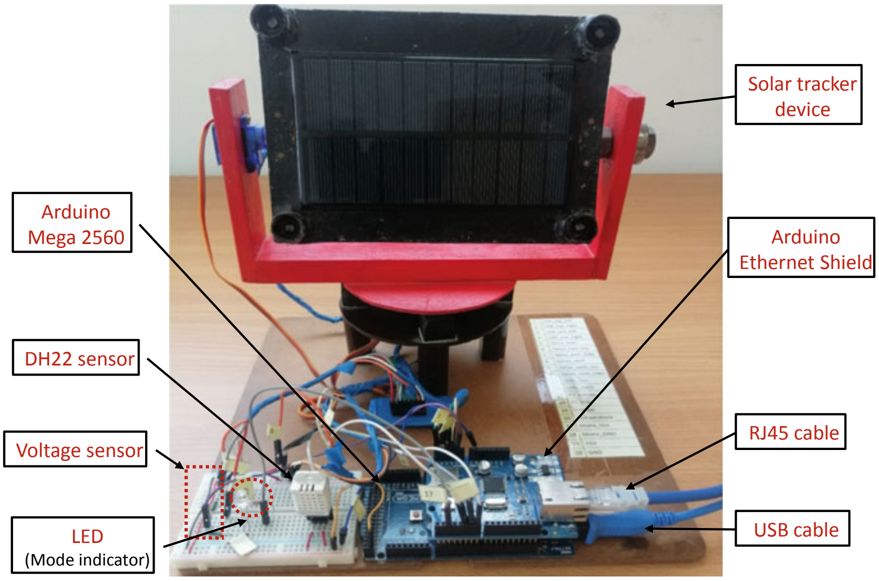


Fig. 5 IoT-based solar tracker prototype

2.4 Software Design

a. Arduino IDE

Arduino is an open-source electronics prototyping platform with easy-to-use hardware and software [32]. The Arduino platform provides an integrated development environment (IDE), which includes support for C and C++ programming languages. The used Arduino board in this work is programmed by the IDE that serves as a code editor and from which the program code can be uploaded to the microcontroller through USB cable, as can be shown in Fig. 3. The Arduino Megaboard is utilized to implement all software requirements of the IoT-based solar tracker.

b. MyDevices Cayenne

myDevices is a company that offers IoT solutions. It offers an end-to-end platform for the IoT. In our project, we will focus on Cayenne, one of the solutions from myDevices. This tool allows developers, designers, and engineers to build prototypes of the IoT. Cayenne uses the Message Queuing Telemetry Transport (MQTT) protocol to connect any device with the Cayenne cloud. Once connected, the user can send and receive data from the device to the Cayenne dashboard via the Widgets created. MQTT is a publish–subscribe messaging protocol based on the TCP/IP protocol. The publish–subscribe methodology uses a message agent that is responsible for delivering messages to the client. The MQTT is the API for sending information to the Cayenne cloud, or devices controlled by Cayenne. The messaging agent in this connection is the cloud, it manages the different clients (sensors and actuators) that send and receive the data.

To use MQTT with Cayenne, we need to use the Cayenne libraries. For Arduino, the CayenneMQTT library can be installed from the IDE's Library Manager. To program our Cayenne IoT platform-based IoT application, we will take advantage of the predefined functions. For example, to establish the connection between Cayenne cloud and Arduino Mega equipped with the Ethernet module, we call the CayenneMQTT Ethernet library where we declare our authentication information (the username, password and the ClientID) which should be obtained from the Cayenne Dashboard. Then, in the setup part of the program, we call *Cayenne.begin ()* function to establish the connection with Cayenne dashboard. For each actuator, we create a function with an integer parameter between 0 and 31 imperatively called *CAYENNE IN (VIRTUAL CHANNEL)*. For each sensor, we create a function with an integer parameter between 0 and 31 imperatively called *CAYENNE_OUT (VIRTUAL_CHANNEL)*. In the loop part of the program, we call the predefined function *Cayenne.loop ()*, this function itself calls the functions *CAYENNE_OUT* and *CAYENNE_IN*. The virtual channel as its name suggests is a channel that does not physically exist, it characterizes visualization or command widgets. It allows them to be linked with the corresponding sensor or actuator.

c. The Embedded Software Design

The embedded software is the piece that will be embedded in the Arduino Megato interact between the Ethernet module and Cayenne cloud (see Appendix). It is designed as follow:

(i) The IoT-based solar tracker has two function modes: manual and automatic. A button created in the Cayenne dashboard has a role to switch between the two modes. When it is inactive, the manual mode is selected, otherwise automatic mode. Besides, a function is established in the Arduino code that allows recovering the state of the button. The LED in the system circuit reflects the state of this switch.

Therefore, for the controller to know the selected operating mode, we just need to test the state of the pin in which the LED is connected. For example, if the LED state is low, the controller will call the manual mode function to execute, otherwise, it will call the automatic function.

(ii) If the manual mode is selected, the user can directly control the positions of the servomotors to orient the PV panel from east to west by L-R servomotor or from south to north by the U-D servomotor. The control is made from the associated widgets of servomotors in the dashboard of the IoT application.

In this mode, the controller calls *Cayenne.loop ()* function which itself calls all the functions *CAYENNE_IN*, including those related to servomotors, to execute. The Cayenne.loop () function will also call all the functions CAYENNE_OUT, linked to the sensors, to execute. Where the data related to LDR sensors, PV current, voltage and power, temperature and humidity would be sent to the server so that they can be visualized in their associated widgets in the IoT application.

(iii) If the automatic mode is selected, the algorithm shown in Fig. 6 will be executed. The algorithm starts by reading the analog values returned by LDR sensors. Then, it processes these data to command servomotors that move the PV panel toward the sun position. Considering the vertical axis-based solar tracker movement, the average values of the two LDRs on the left and the two LDRs on the right are compared and if the lefts receive more light, the PV panel will move in that direction (clockwise) through the L-R servomotor. The latter will stop when the difference result is between -10 and 10. This range is used to stabilize the controller and to reduce the power consumption of servomotors. Otherwise, if the right set of LDRs receives more light, the PV panel will move in that direction (Counterclockwise) through the L-R servomotor and will continue to rotate until the difference result is in the range $[-10, 10]$. The same approach is used for the horizontal axis-based solar tracker movement where the average values of the two LDRs on the top and the two LDRs on the bottom are compared.

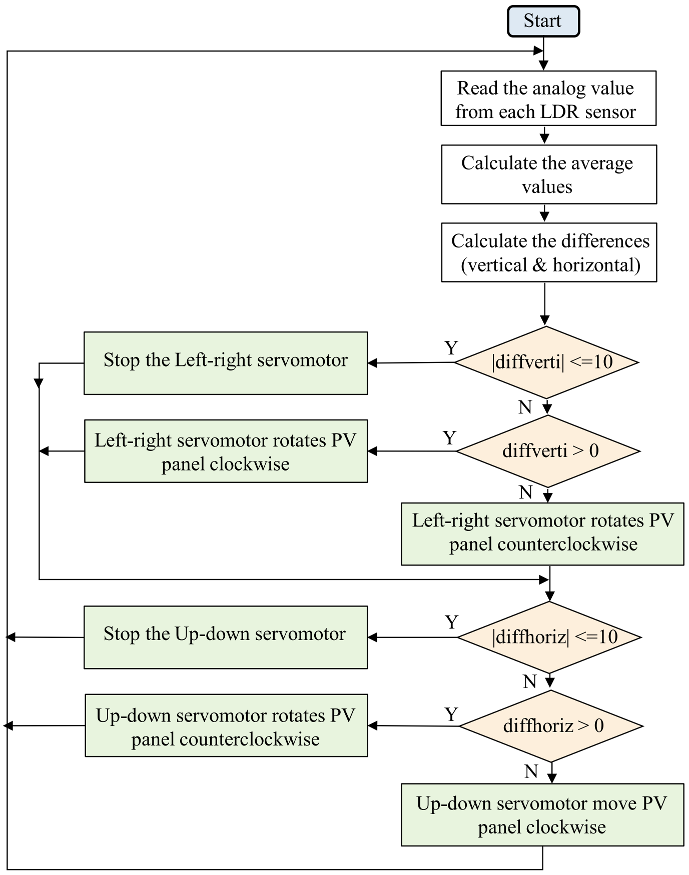

Start
Read the analog value from each LDR sensor
Calculate the average values
Calculate the differences (vertical & horizontal)
|diffverti| <=10
Y
N
Stop the Left-right servomotor
diffverti > 0
Left-right servomotor rotates PV panel clockwise
Left-right servomotor rotates PV panel counterclockwise
|diffhoriz| <=10
Stop the Up-down servomotor
diffhoriz > 0
Up-down servomotor rotates PV panel counterclockwise
Up-down servomotor move PV panel clockwise

Fig. 6 The flowchart for the automatic mode of the solar tracker

As well as in the automatic mode, the controller will also call the *Cayenne.loop ()* function to send the solar tracker data to the IoT application.

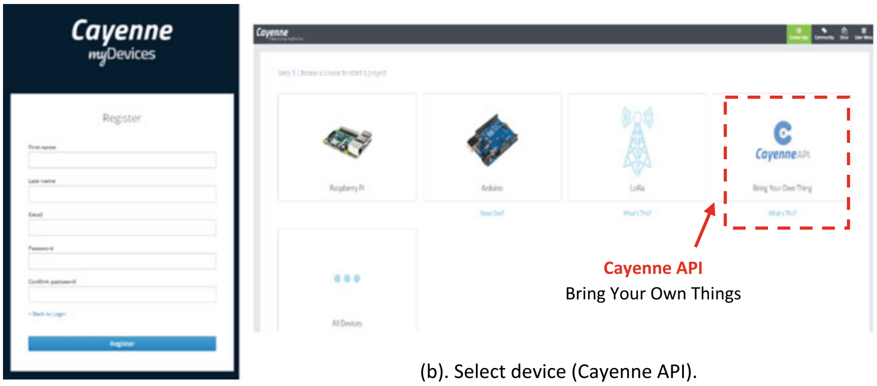
For

d. Development of the IoT Monitoring Application

(i) *Hardware interfacing with Cayenne IoT platform*

To interface the hardware, including sensors and actuators, with the IoT platform, we need to follow the next steps:

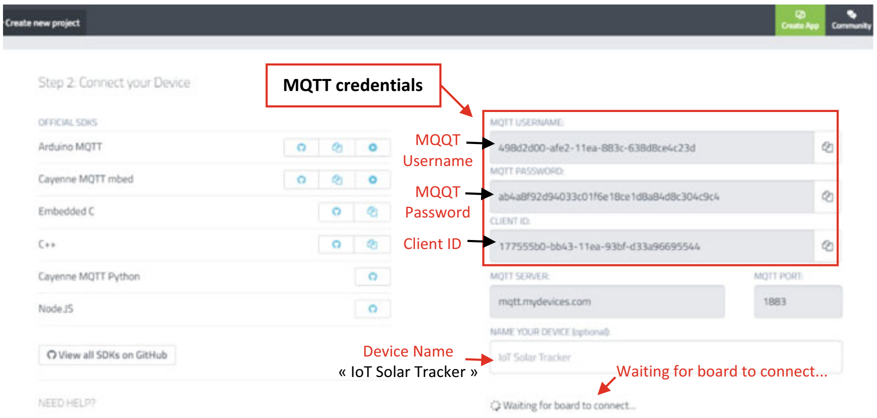
- Log in on Cayenne myDevice website after creating an account (Fig. 7a).
- Then, click on “Bring Your Own Things” from Cayenne API (Fig. 7b).
- Copy the MQTT credentials (username, password and client ID) from Create App (Fig. 8), and paste them in Arduino source code as described previously. After successfully compiling and uploading the entire code to Arduino Mega, open



(a). Cayenne sign. up

(b). Select device (Cayenne API).

Fig. 7 Cayenne IoT Platform sign up (**a**). Cayenne API (**b**)

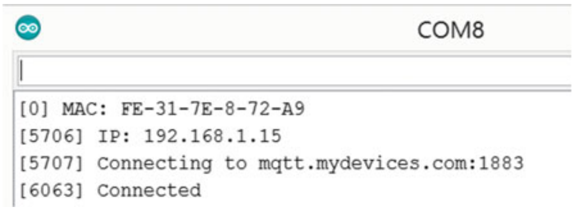


MQTT USERNAME: 498d2d00-afe2-11ea-883c-638d8ce4c23d
MQTT PASSWORD: ab4a8f92d94033c01f6e18ce1d8a84d8c304c9c4
CLIENT ID: 177555b0-bb43-11ea-93bf-d33a96695544
MQTT SERVER: mqtt.mydevices.com
MQTT PORT: 1883
NAME YOUR DEVICE (optional): IoT Solar Tracker
Waiting for board to connect...

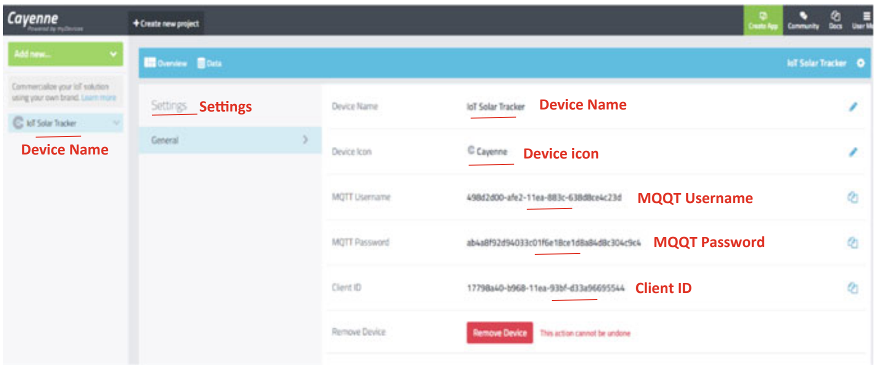
Fig. 8 MQTT credentials and device connection to Cayenne

Serial Monitor in Arduino IDE to get the Cayenne log prints (Fig. 9). As soon as our device comes online and connects to Cayenne, the previous page (Fig. 8) is automatically updated and we will see our device in the online dashboard as can be seen in Fig. 10.

Fig. 9 Cayenne log prints on Serial Monitor



```
[0] MAC: FE-31-7E-8-72-A9
[5706] IP: 192.168.1.15
[5707] Connecting to mqtt.mydevices.com:1883
[6063] Connected
```



Device Name: IoT Solar Tracker
Device Icon: Cayenne
MQTT Username: 498d2d00-afe2-11ea-883c-638d8ce4c23d
MQTT Password: ab4a8f92d94033c01f6e18ce1d8a84d8c304c9c4
Client ID: 17798a40-b968-11ea-93bf-d33a96695544
Remove Device: Remove Device — This action cannot be undone

Fig. 10 Device settings

- Then, to interface sensors and actuators, i.e. create their widgets, click on "Add new…", select "Device/Widget" and click on "Custom Widgets" (Fig. 11). Then, select a widget and populate all its associated settings (the channel number must be the same as in code), and finally click on "Add Widget" to add it to the dashboard of your device. For us, we chose the "value" widget for all sensors, "Button" widget for mode switch and the "Slider" widget for servomotors.

Finally, Fig. 12 illustrates the designed IoT application for monitoring solar tracker data. Once the connection with the solar tracker system is established, sensor data can be visualized on their associated widgets, the tracking mode (automatic or manual) can be selected from the switch button, as well as controlling servomotors' angles through their widgets. Sensor data can also be obtained in graphical form by modifying the representation type in their settings, or just by clicking on the graph icon above the widget.

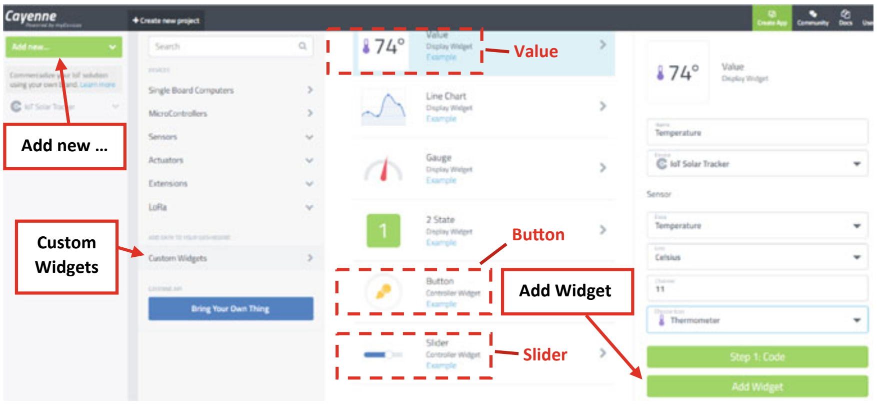
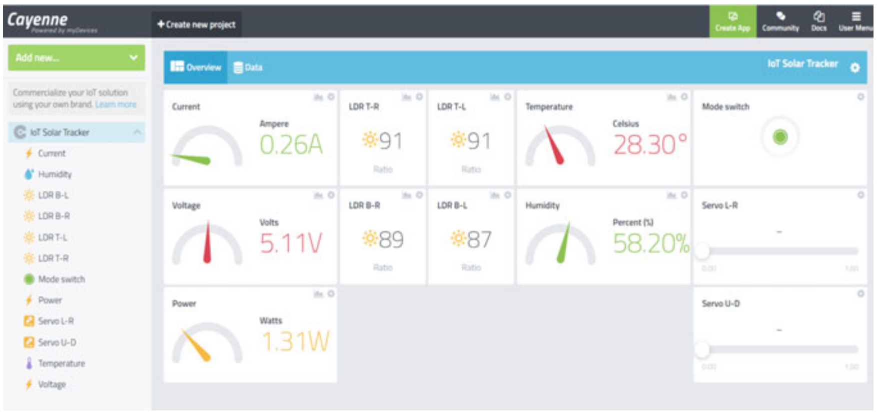


Fig. 11 Cayenne custom widgets

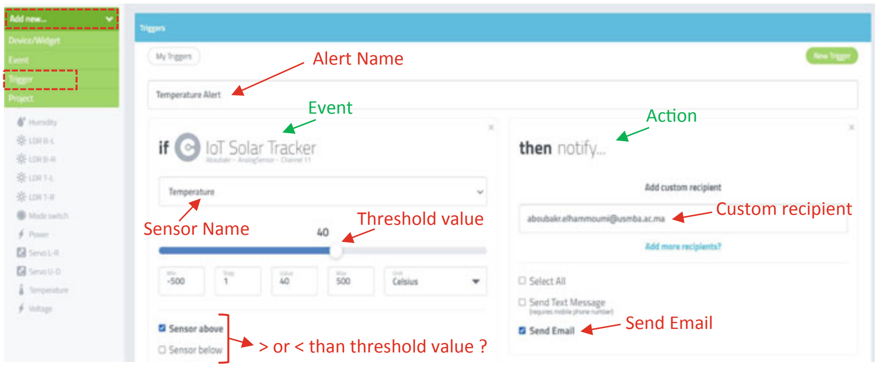


Widget	Unit	Value
Current	Ampere	0.26A
LDR T-R	Ratio	91
LDR T-L	Ratio	91
Temperature	Celsius	28.30°
Voltage	Volts	5.11V
LDR B-R	Ratio	89
LDR B-L	Ratio	87
Humidity	Percent (%)	58.20%
Power	Watts	1.31W

Fig. 12 IoT monitoring application of solar tracker system

(ii) *Alerts creation*

One of the most important criteria in a monitoring system is its ability to send notification alerts to inform users when an event, related to their monitored devices, occurs. To this end, we take advantage of one of Cayenne's features [33] to add alerts to our IoT application, where we can preprogram our application to send a notification alert (SMS, Email, or both) or to perform a specified action. For example, a temperature alert is created to send an email notification to the user (or recipients) when the monitored temperature is reached a threshold value, as can be shown in Fig. 13. To create an alert, click on "Add new…" and select "Trigger", then set the event and its action and finally click on "save" to add it to the dashboard.



Temperature Alert
Temperature
40
Min -500 | Step 1 | Value 40 | Max 500 | Unit Celsius
[✓ Sensor above] Sensor below
Add custom recipient
aboubakr.elhammoumi@usmba.ac.ma
Add more recipients?
Select All
Send Text Message
[✓ Send Email]

Fig. 13 Temperature alert configuration

3 Results and Discussion

Different tests have been carried out to examine the developed IoT-based solar tracker prototype. The experimental setup is illustrated in Fig. 14. The Arduino board is powered with the computer through a USB cable, which is also used to display, in the Serial Monitor of Arduino IDE, the measured parameters and data received from the IoT application. This will allow us to verify whether the captured data from Arduino are correctly and in real-time sent to the IoT application or not. Whereas, Arduino can be powered with an external DC power supply. The Ethernet shield connects the Arduino board to the internet via RJ45 cable. Once the connection with the IoT application is established, the data of the solar tracker system are sent to the monitoring application, where we can view these data live and send commands to the controller.

The system is programmed to send all data from the device regardless of the tracker mode (manual or automatic). First, the automatic mode has been tested, which is activated when the switch mode is in the high state; the LED (mode indicator) in the circuit lights up. The servomotors were automatically controlled according to intensities captured by LDR sensors. Figure 15 presents the samples of data recorded in real time from the solar tracker system in the IoT application during the test period. Figure 15a displays the recorded electrical measurements, namely the current, voltage and power. While Fig. 15b displays the recorded environmental measurements, namely the temperature, humidity and the intensity of light (captured by the top-right LDR) with the accurate time and date. It has been verified, by comparing the data sent from Arduino and those received on the dashboard of the

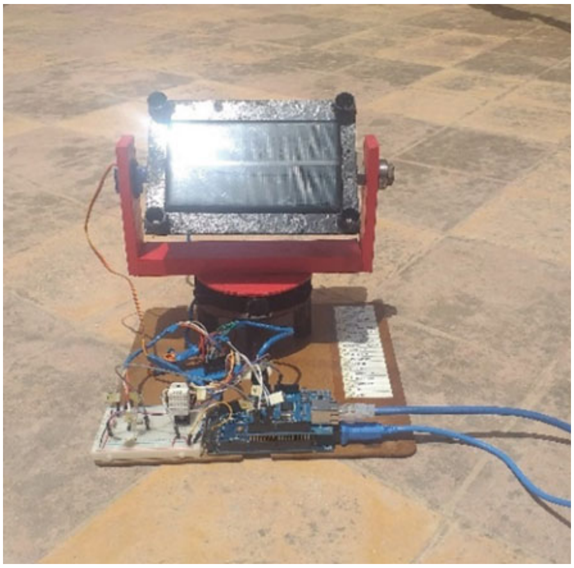
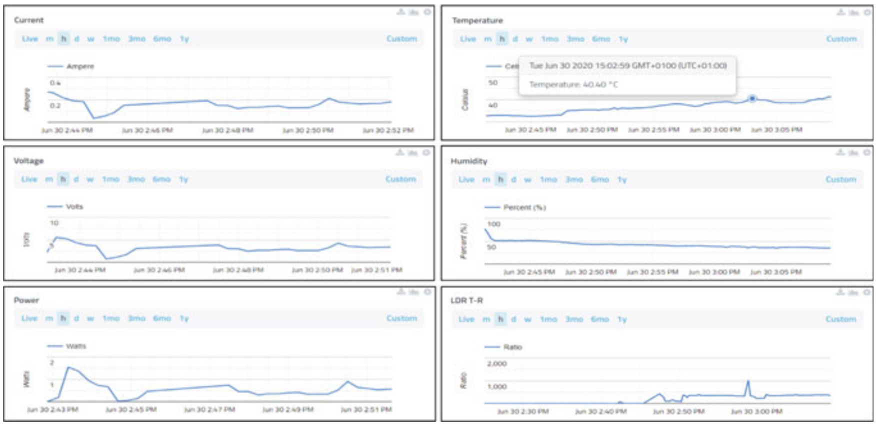


Fig. 14 The experimental setup of the prototype



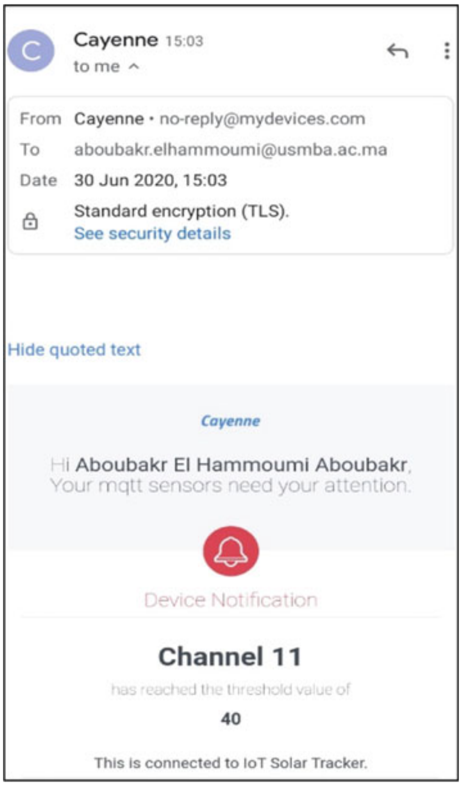
(a) Electrical measurements. (b). Environmental measurements.

Fig. 15 Samples of data recorded in real time in the IoT application

IoT application, that all electrical and environmental measurements are sent properly and in real-time.

To check the reliability of the monitoring application to notify the user when an event occurs, it has been programmed to send an alert. For example, when the monitored temperature is higher than 40 °C. Figure 16 shows the alert notification received in our mailbox at the same time when the temperature exceeds 40° as can be seen in Fig. 15b. Other alerts can be added to the application, such as a malfunction of one of the sensors and/or actuators and a rapid decrease in PV power.

Moreover, the proposed IoT prototype has been tested in manual mode, which is activated when the switch mode is in the low state; the LED in the circuit turns off. The servomotors were controlled through their associated widgets in the dashboard. For instance, we have set the slider linked to the L-R servomotor at the center (i.e. at a value of 0.5) and the other slider of U-D servomotor at 0.3, which means that the L-R and U-D motors will rotate by 90° and 30°, respectively. Also, tests have shown that solar tracker properly and rapidly executes commands from the monitoring platform with a time not exceeding 2 s. In the manual mode, the user can remotely position his device in an optimal direction according to the surrounding environment and device location. Moreover, in this mode, the power consumption of motors can be too minimized or, where the user can intervene to position the solar tracker for example in only one direction according to each month or season (winter, spring, summer, and autumn) in the year. However, an amount of PV energy can be lost due to a limitation of the solar tracker operation according to the daily movement of the sun. The servomotor commands can be programmed beforehand without user intervention by creating events and associated actions in the IoT application.


Cayenne 15:03
to me

From Cayenne • no-reply@mydevices.com
To aboubakr.elhammoumi@usmba.ac.ma
Date 30 Jun 2020, 15:03
Standard encryption (TLS).
See security details

Hide quoted text

Cayenne

Hi Aboubakr El Hammoumi Aboubakr,
Your mqtt sensors need your attention.

Device Notification

Channel 11
has reached the threshold value of
40

This is connected to IoT Solar Tracker.

Fig. 16 The received notification alerts in the mailbox

4 Conclusion

In this chapter, a smart prototype has been designed to monitor and control a dual-axis solar tracker system using a simple and efficient IoT solution. The prototype has been tested experimentally. Test results demonstrate that the developed IoT-based solar tracker provides users with a simple monitoring application, in which users can easily and in real-time monitor electrical and environmental parameters of the solar tracker system for further processing and management. Other sensors could be added, for example, solar irradiation and wind sensors to help understand more about the PV power output as well as to test the solar tracker system on its flexibility during high wind. Due to its simplicity, the proposed IoT solution can be employed in various fields to connect devices or things to the internet as well as for research or educational purposes.

Appendix

The embedded code of the IoT-based solar tracker system

```
/****************************************************************
 PROJECT: IoT based solar tracker system / the embedded software
****************************************************************/

#define CAYENNE_PRINT Serial
#include <CayenneMQTTEthernet.h>    //CayenneMQTT library
#include <Servo.h>                  //Servo motor library
#include <DHT.h>                    //DHT library
#define DHTTYPE DHT22
#define DHTPIN 2
DHT dht(DHTPIN,DHTTYPE);

//MQTT credentials
char username[]="498d2d00-afe2-11ea-883c-638d8ce4c23d";
char password[]="ab4a8f92d94033c01f6e18ce1d8a84d8c304c9c4";
char clientID[]="17798a40-b968-11ea-93bf-d33a96695544";

Servo servo_x;                    //up-down servomotor
int servoh = 0;
int servohLimitHigh = 170;
int servohLimitLow = 10;

Servo servo_z;                    //left-right servomotor
int servov = 0;
int servovLimitHigh = 170;
int servovLimitLow = 10;

int topl,topr,botl,botr;
int threshold_value=10;
float Vout;

void setup()
{ Serial.begin(9600);
  Cayenne.begin(username, password, clientID);
  servo_x.attach(5);
  servo_z.attach(6);
  dht.begin();
  pinMode(3,OUTPUT);
  digitalWrite(3,LOW);
}

void loop()
{ topr= analogRead(A2);
  topl= analogRead(A3);
  botl= analogRead(A4);
  botr= analogRead(A5);
  Vout=(analogRead(A1) * 5.0) / 1023;
  Serial.println(" Manual-mode");
  Cayenne.loop();

  if (digitalRead(3)==HIGH){
    Serial.println(" Automatic-mode");
    servoh = servo_x.read();
    servov = servo_z.read();
    int avgtop = (topr + topl) / 2;
    int avgbot = (botr + botl) / 2;
    int avgright = (topr + botr) / 2;
    int avgleft = (topl + botl) / 2;
    int diffhori= avgtop - avgbot;
    int diffverti= avgleft - avgright;

    /*tracking according to horizontal axis*/
    if (abs(diffhori) <= threshold_value)
    {
     servo_x.write(servoh);             //stop the servo up-down
    }else {
       if (diffhori > threshold_value)
          { Serial.println(" x - 2 ");
          servo_x.write(servoh -2);     //Clockwise rotation CW
          if (servoh > servohLimitHigh)
          {
           servoh = servohLimitHigh;
          }
          delay(10);
          }else {
           servo_x.write(servoh +2);    //CCW
           if (servoh < servohLimitLow)
           {
           servoh = servohLimitLow;
           }
           delay(10);
           }
      }
    /*tracking according to vertical axis*/
    if (abs(diffverti) <= threshold_value)
    {
     servo_z.write(servov);       //stop the servo left-right
    }else{
       if (diffverti > threshold_value)
       {
       servo_z.write(servov -2);  //CW
       if (servov > servovLimitHigh)
       {
       servov = servovLimitHigh;
       }
       delay(10);
       }else{
        servo_z.write(servov +2);  //CCW
        if (servov < servovLimitLow)
        {
        servov = servovLimitLow;
        }
        delay(10);
        }
     }
  }
}
// Cayenne Functions
CAYENNE_IN(8){
  int value = getValue.asInt();
  CAYENNE_LOG("Channel %d, pin %d, value %d", 8, 3, value);
  digitalWrite(3,value);
}
CAYENNE_IN(7){ //up-down servo motor
  if (digitalRead(3)==HIGH){ //Automatic_mode
  }
  else{ //Manual_mode
  servo_x.write(getValue.asDouble() * 180);
  }
}
CAYENNE_IN(6){ //left-right servo motor
  if (digitalRead(3)==HIGH){
  }
  else{
  servo_z.write(getValue.asDouble() * 180);
  }
}
```

```
CAYENNE_OUT(0) { //Current
  float current = Vout/10;
  Cayenne.virtualWrite(0, current);
  Serial.print("Current: ");
  Serial.println(current);
}
CAYENNE_OUT(1) { //Voltage
  float voltage = Vout * 2;
  Cayenne.virtualWrite(1, voltage);
  Serial.print("Voltage: ");
  Serial.println(voltage);
}
CAYENNE_OUT(2){ //LDR Top-right
  Cayenne.virtualWrite(2, topr);
}
CAYENNE_OUT(3){ //LDR Top-left
  Cayenne.virtualWrite(3,topl);
}
CAYENNE_OUT(4){ //LDR Bot-left
  Cayenne.virtualWrite(4,botl);
}
CAYENNE_OUT(5){ //LDR Bot-right
  Cayenne.virtualWrite(5,botr);
}
CAYENNE_OUT(10) { //Power
  float power = (Vout * 2 * Vout)/10 ;
  Cayenne.virtualWrite(10, power);
  Serial.print("Power: ");
  Serial.println(power);
}
CAYENNE_OUT(11){ //Temperature
  float t = dht.readTemperature();
  //int chk = dht.read(DHT11PIN);
  Cayenne.virtualWrite(11, t, TYPE_TEMPERATURE, UNIT_CELSIUS);
  Serial.print("temperature: ");
  Serial.println(t);
}
CAYENNE_OUT(12){ //Huidity
  float h = dht.readHumidity();
  //int chk = dht.read(DHT11PIN);
  Cayenne.virtualWrite(12, h);
  Serial.print("  humidity: ");
  Serial.println(h);
}
```

References

1. International Telecommunications Union (2012) Overview of the internet of things
2. Kang B, Kim D, Choo H (2017) Internet of everything: a large-scale autonomic iot gateway. IEEE Trans Multi Scale Comput Syst 3(3):206–214
3. Ahmed MA, Eltamaly AM, Alotaibi MA, Alolah AI, Kim YC (2020) Wireless network architecture for cyber physical wind energy system. IEEE Access 8:40180–40197
4. Business Insider Intelligence (2020) The internet of things report
5. Haghi M, Thurow K, Stoll R (2017) Wearable devices in medical internet of things: Scientific research and commercially available devices. Healthc Inform Res 23(1):4–15
6. Jia M, Komeily A, Wang Y, Srinivasan RS (2019) Adopting internet of things for the development of smart buildings: a review of enabling technologies and applications. Autom Constr 101:111–126

7. Khajenasiri I, Estebsari A, Verhelst M, Gielen G (2017) A review on internet of things solutions for intelligent energy control in buildings for smart city applications. Energy Procedia 111:770–779
8. YIN Y, Zeng Y, Chen X, Fan Y (2016) The internet of things in healthcare: an overview. J Indust Inform Integr 1(Elsevier B.V):3–13
9. Tzounis A, Katsoulas N, Bartzanas T, Kittas C (2017) Internet of things in agriculture, recent advances and future challenges. Biosyst Eng 164:31–48. (Academic Press)
10. Li JQ, Yu FR, Deng G, Luo C, Ming Z, Yan Q (2017) Industrial internet: a survey on the enabling technologies, applications, and challenges. IEEE Commun Surv Tutorials 19(3):1504–1526
11. Rahman MM, Selvaraj J, Rahim NA, Hasanuzzaman M (2018) Global modern monitoring systems for PV based power generation: a review. Renew Sustain Energy Rev 82:4142–4158. (Elsevier Ltd)
12. Nsengiyumva W, Chen SG, Hu L, Chen X (2018) Recent advancements and challenges in Solar Tracking Systems (STS): a review. Renew Sustain Energy Rev 81:250–279. (Elsevier Ltd)
13. Hafez AZ, Yousef AM, Harag NM (2018) Solar tracking systems: Technologies and trackers drive types − a review. Renew Sustain Energy Rev 91:754–782. (Elsevier Ltd)
14. Sumathi V, Jayapragash R, Bakshi A, Kumar Akella P (2017) Solar tracking methods to maximize PV system output − a review of the methods adopted in recent decade. Renew Sustain Energy Rev 74:130–138. (Elsevier Ltd)
15. El Hammoumi A, Motahhir S, El Ghzizal A, Chalh A, Derouich A (2018) A simple and low-cost active dual-axis solar tracker. Energy Sci Eng 6(5):607–620
16. Motahhir S, EL Hammoumi A, EL Ghzizal A, Derouich A (2019) Open hardware/software test bench for solar tracker with virtual instrumentation. Sustain Energy Technol Assessm 31:9–16
17. AL-Rousan N, Isa NAM, Desa MKM (2018) Advances in solar photovoltaic tracking systems: a review. Renew Sustain Energy Rev 82:2548–2569. (Elsevier Ltd)
18. Nur M, Mohd A, Jumaat A, Rimong C, Jawa A, Jumaat SA (2020) Dual axis solar tracker with IoT monitoring system using arduino. Int J Power Electron Drive Syst 11(1):451–458
19. Pulungan AB, Risfendra R, Purwanto W, Maksum H, Setiawan O (2020) Design and development of real time monitoring single axis solar tracker by using internet of things
20. Williams K, Qouneh A (2017) Internet of things: solar array tracker. Midwest Symp Circuits Syst 2017:1057–1060
21. Singh KJ, Kapoor DS (2017) Create your own internet of things: a survey of IoT platforms. IEEE Cons Electron Mag Inst Electr Electron Eng Inc 6(2):57–68
22. Lopez-Vargas A, Fuentes M, Vivar M (2019) IoT application for real-time monitoring of solar home systems based on arduinotm with 3G connectivity. IEEE Sens J 19(2):679–691
23. Mekala MS, Viswanathan P (2019) CLAY-MIST: IoT-cloud enabled CMM index for smart agriculture monitoring system. Meas J Int Meas Conf 134:236–244
24. Amir Alavi S, Rahimian A, Mehran K, Alaleddin Mehr Ardestani J (2018) An IoT-based data collection platform for situational awareness-centric microgrids. In: Canadian Conference on Electrical and Computer Engineering, vol 2018
25. Motahhir S et al (2020) Optimal energy harvesting from a multistrings PV generator based on artificial bee colony algorithm. IEEE Syst J 1–8
26. El Hammoumi A, Motahhir S, Chalh A, El Ghzizal A, Derouich A (2018) Real-time virtual instrumentation of Arduino and LabVIEW based PV panel characteristics. IOP Conf Ser Earth Environ Sci 161(1):012019
27. El Hammoumi A, Motahhir S, Chalh A, El Ghzizal A, Derouich A (2018) Low-cost virtual instrumentation of PV panel characteristics using Excel and Arduino in comparison with traditional instrumentation. Renew Wind Water Sol 5(1):1–16
28. Cayenne-myDevices (2020). https://mydevices.com/. (Accessed 23 June 2020)
29. V/1.6 W Small Solar Panel (2020). https://fr.enfsolar.com/pv/panel-datasheet/crystalline/34434. (Accessed 23 June 2020)

30. DHT22 Sensor Datasheet (2020). https://datasheetspdf.com/pdf/792211/Aosong/DHT22/1. (Accessed 23 June 2020)
31. W5100 | WIZnet Co., Ltd. (2020). https://www.wiznet.io/product-item/w5100/. (Accessed 23 June 2020)
32. Zlatanov N (2020) Arduino and open source computer hardware and software
33. myDevices, Cayenne Docs (2020). https://developers.mydevices.com/cayenne/docs/features/#features-alerts. (Accessed 10 June 2020)