Electric Measuring System Documentation

Pi-Sense

14/06/2023

1 Team members

1. Lochana Abhayawardana
2. Emirkan Sali
3. Arsany Girgis

2 Introduction

The Electricity Measuring System is a prototype designed to measure and observe current, voltage and the temperature of an industrial factory or a lab. As a safety feature the prototype is capable of give alerts/warning system and take the necessary safety actions. The user can keep a track on the system with a mobile device or a laptop. In case of emergency the user is capable of terminate the malfunctioning/ failing sections of the electrical system remotely.

In order to gain the previously explained capabilities, it is required to implement a wireless sensor network. The Raspberry pi 3 players a as the central system and the data processing and transfer unit of the prototype. MQTT (Message queuing Telemetry Transport) protocol is used to transfer signal wirelessly. The current sensor, voltage sensor and the temperature sensor has connected to an Arduino Wi-Fi Rev 2 which act as the nodes of the wireless sensor network. This Arduino work as a publisher to the broker to transfer the measurements.

This system is implemented by using Internet of things. Raspberry pi is connected to the internet and use the cloud technology to process and store measurements from the sensors, record of a publishers, commands etc.

3 Concept description

In the Block diagram in figure 1 we can see the general structure of our electricity measuring wireless sensor network. We will use a Raspberry Pi as a broker for all our subscriber and publisher devices. In this concept block diagram, we have 2 Arduino microcontrollers which act as publishers. Each publisher device is controlling and gathering information from a voltage sensor, a current sensor and a temperature sensor. This information will be distributed to all subscribers of the network. The Arduino device, which acts as a subscriber is controlling the actuators of the system by using the information from the sensors and cloud service. The actuators in this case are a Door Lock/motor device for locking the door, Green and red LEDs and a buzzer or any actuator that will create an alarm sound. Additionally, all information from the sensors is being displayed for the Computer/Phone Subscriber devices for the users of the system to check on the measured values.

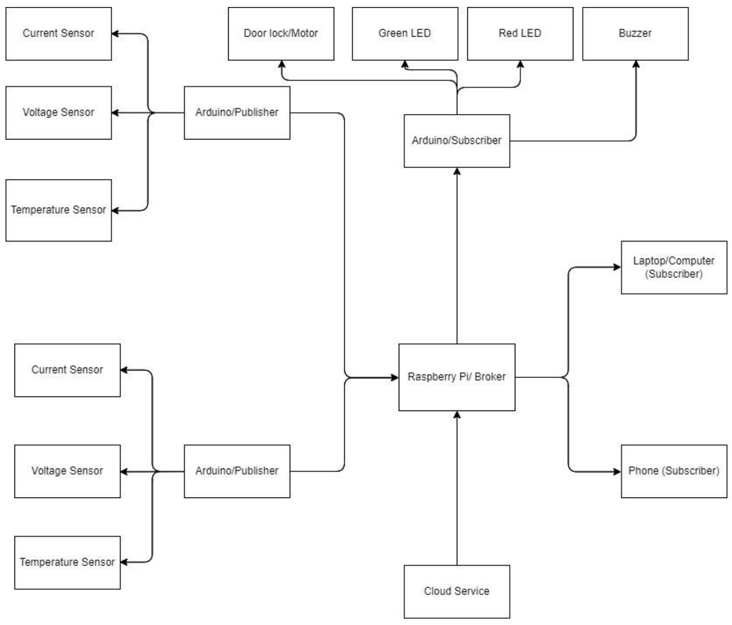


Figure 1: Block Diagram for our wireless sensor network

The main application for our system is in an industrial testing environment in which high voltage or low voltage systems are tested for their functionalities. For high voltage applications our system is designed to measure voltages and currents in internal circuits of the test device and not the applied high voltage, for example a control or driver board of an inverter. Additionally, our WAN will provide a way for the users to measure the temperature on certain points of the test device and see the information on one of the subscriber devices.

In high voltage applications, safety measures are to be taken, so that no one can touch the test device while it is under high voltage, thus our System includes a door lock to prevent any person from entering the test environment after setting up the sensors. Furthermore, the system provides ways of reacting to a voltage, current or temperature above a set threshold, in which the system will alert the user using LEDs and a Buzzer/alarm device.

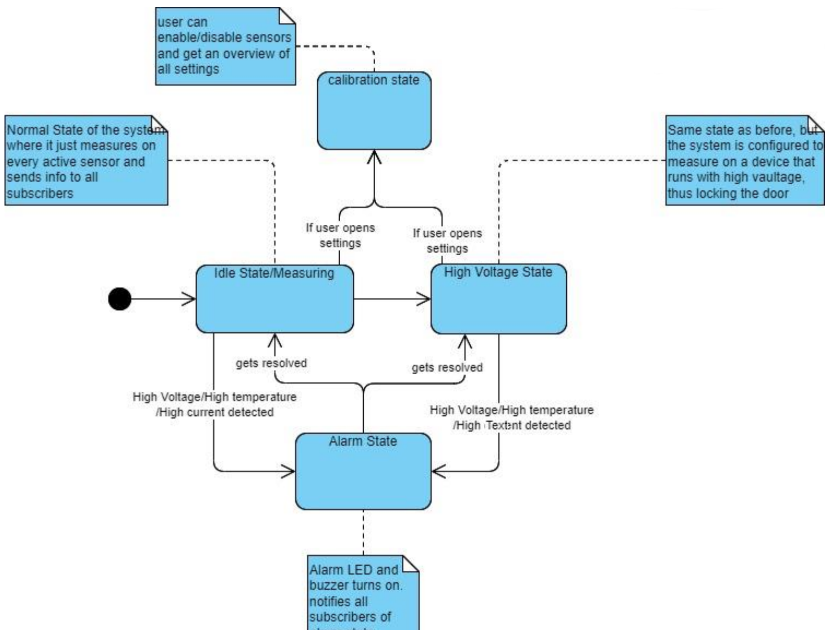

user can enable/disable sensors and get an overview of all settings
calibration state
Normal State of the system where it just measures on every active sensor and sends info to all subscribers
Same state as before, but the system is configured to measure on a device that runs with high vaultage, thus locking the door
If user opens settings
If user opens settings
Idle State/Measuring
High Voltage State
gets resolved
gets resolved
High Voltage/High temperature /High current detected
High Voltage/High temperature /High Textent detected
Alarm State
Alarm LED and buzzer turns on. notifies all subscribers of

Figure 2: State diagram of our System concept

4 Project/Team management

In the project, we mostly worked on all tasks together during the labs and on meetup occasions during the week, but team members had a specific focus on some tasks nevertheless:

- *Lochana: Setting up the raspberry pi by installing Raspbian os and enabling MQTT as well as creating a sequence diagram for the concept and helping with setting up the sensors and actuators.*

- *Arsany: Set up of the raspberry Pi Mosquitto module and enabling the communication via MQTT and ultimately setting up the Raspberry Pi as a broker. Created the Activity diagram.*
- *Emirkan: Focus on writing arduino code for actuator and sensor devices. Creating a state, class and block diagram for the system. Implementation of the Publisher and subscriber connections of the Arduino devices. Wrote all sections of the documentation.*

5 Technologies

For the project we used 2 Arduino Uno Wi-Fi Micro-controllers, a raspberry Pi and several actuators and sensors. We used one main programming language and one main communication protocol for wireless data exchange. Additionally for user Feedback, the mobile app “IoT MQTT Panel” was used [4] and the desktop program “MQTTX” [5].

5.1 Communication Protocol

The MQTT messaging protocol is the main communication protocol used in our project. MQTT means MQ Telemetry Transport and it is mainly used for IoT or industrial IoT devices to exchange data by publishing data to certain topics and subscribing to these topics to receive the published data. This data exchange can be between several sensors, actuators or embedded devices. In our project’s case, we used the communication protocol to establish communication between our sensor and actuator devices while providing feedback to a subscribed user. The structure of an MQTT network is as follows:

- There are publisher devices that subscribe to one or more topics. Topics can be anything related to the purpose of the exchanged data. They are responsible for differentiating from the different data exchanges and provide the information about the purpose of the published or subscribed data. In our project the sensor devices are the publishers of the MQTT network
- Subscriber devices simply wait for data from the subscribed topic to take as input for many possible purposes. In our Project the actuator devices subscribe to the topics published by the sensors to provide a way to react to some sensor values. Similarly feedback of the sensor values for the Users is provided by subscribing to the topic
- The broker handles the connection between the subscriber and the publisher devices by filtering all incoming messages and distributing them accurately to the proper subscribers. In our project we used a RaspBerry Pi with the mosquito module installed as a broker.

5.2 Sensors

1. Current Sensor: For current measurements, we used the fully integrated, hall-effect-based linear current sensor ACS712 by Allegro microsystems. We simply provide 5V operation voltage to the sensor from the Arduino microcontroller and attach one signal pin to one of the analog pins of our Arduino. The sensor has an input and output for measuring the current. The current is converted into a voltage at the output signal pin and is converted into a digital signal at the analog pin of the Arduino using its internal ADC. The resulting current in Ampere is computed in mV steps [1].

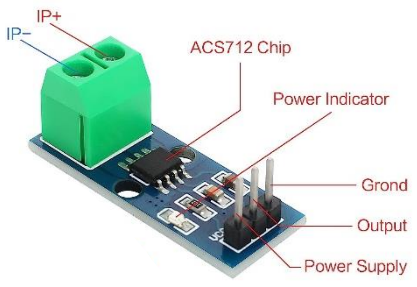


Figure 3: Current Sensor ACS712 [2]

2. Temperature Sensor: For temperature measurements a thermistor sensor by RobotLinking is used, which is connected to an analog pin of the Arduino. The voltage divider circuit is on the PCB board of the sensor which allows us to easily connect the sensor.
3. Selfmade voltage sensor: We created a voltage sensor with a range up to 17.5 volts by using a simple voltage divider circuit with voltage input and signal input into an analog pin. Using a formula in the Arduino code, we can determine the voltage on the input pin of the sensor.

Figure 4: Analog Temperature Module [3]

5.3 Actuators

1. Buzzer: A simple buzzer from the standard Arduino kit is used to create an alarm noise in case of critically high voltage, current or temperature

2. A servo motor from the standard Arduino Kit is used to simulate a door lock.

3. 1 red and 1 green LEDs are used to provide feedback about the status of the system and the alarm, when triggered.

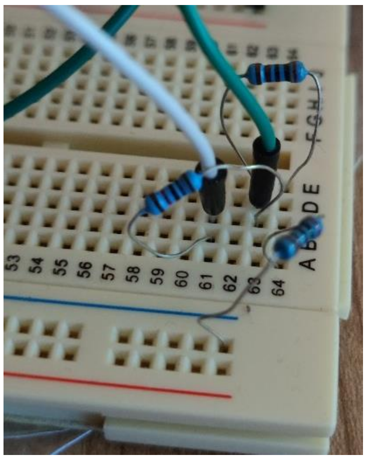


Figure 5: Voltage sensor using a simple voltage divider circuit

5.4 Programming languages

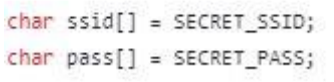
For the main coding of the actuators and the sensors, Arduino micro-controllers are used, therefore the main programming language is the Arduino programming language, which is comparable to C++.

For the initialization of the broker function of the Raspberry Pi, Python programming language was used.

6 Implementation

The project has been implemented as described in the concept on 2 Arduino micro-controllers and one Raspberry Pi. In general, we have 1 Micro-controllers controlling the sensors and a different one controlling the actuators.

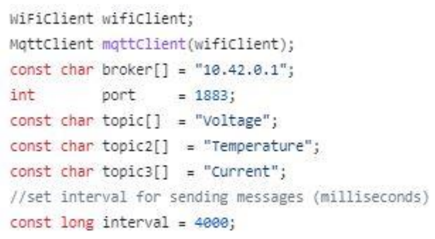
In the Arduino Code for both devices, we import the libraries “WiFiNINA” and “ArduinoMqttClient”, for the necessary functions for establishing the wifi connection to the Raspberry Pi and the subscription and publishing to the MQTT network. Furthermore, a header file called “arduino_secrets” is imported in which the SSID and password of the Wi-Fi network from the Raspberry Pi is stored more securely.



```
char ssid[] = SECRET_SSID;
char pass[] = SECRET_PASS;
```

Figure 6: Setting the SSID an password of the Wi-Fi

Next, we initialize the Wi-Fi client and the MQTT client by establishing the connection between them and creating variables for the MQTT client for information about the broker, port, topics and the time interval between messages.



```
WiFiClient wifiClient;
MqttClient mqttClient(wifiClient);
const char broker[] = "10.42.0.1";
int        port     = 1883;
const char topic[]  = "Voltage";
const char topic2[]  = "Temperature";
const char topic3[]  = "Current";
//set interval for sending messages (milliseconds)
const long interval = 4000;
```

Figure 7: Arduino code displaying the initialisation of the Wi-Fi Client and MQTT Client with its needed variables

After declaring all the needed variables for the sensors and actuators, we connect to the Wi-Fi of the Raspberry Pi and the MQTT client tries to connect to the broker after being connected to the Wi-Fi. For the sensors we cycle between the sensor functions in the main loop of the code. For the actuators, the device waits for input from the subscribed topic in each cycle of its main loop and depending on the input, it activates the corresponding actuators. The only actuator that does not depend on input of the sensors is the door lock, because its purpose is for locking the door of a test chamber before the tests start if it is a high voltage test. Therefore, the operation status of the servo motor depends on only one bool variable as seen in figure 8. A general class diagram of our system is also displayed in figure 8

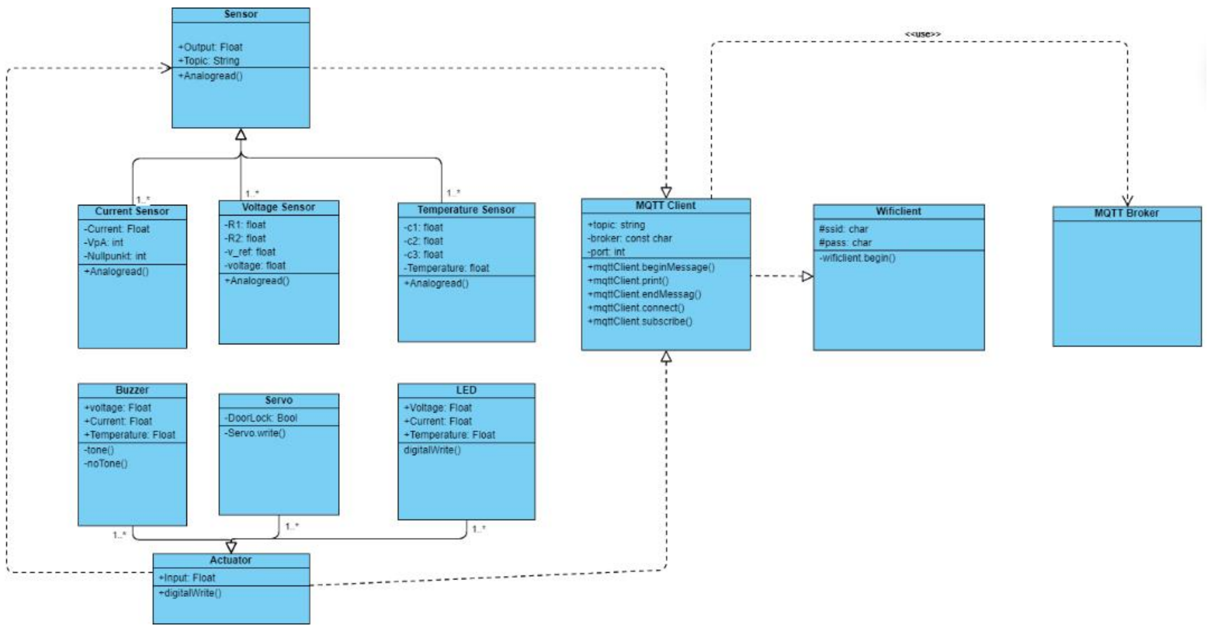


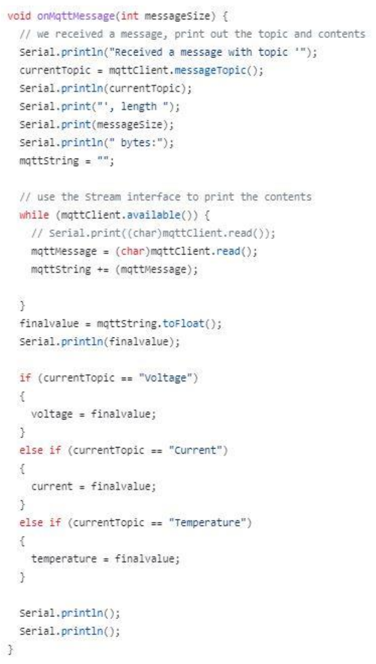
Figure 8: Class diagram of our system

The class diagram provides a general overview of the implementation of each module in the system. The parent classes “Sensor” and “Actuator” both have 3 separate subclasses representing the actuators and sensors of the system. Every sensor provides an output variable which is sent to the Actuator classes as an Input for those. To send the data to another device, the functionality of the “MQTT Client” is needed, which again uses the “WifiClient”, both of which are classes of their own with their provided operation sets.

Every Sensor uses the “Analogread()” function of the arduino, because every sensor used in our system inputs its data through the analog pins. The variable “VpA“ denotes the millivolts per ampere that are to be measured at the analog pin and is set according to the version of the sensor. We use the current sensor with 30 amperes of range, so it is set to 66 according to its datasheet. The “Nullpunkt” variable simply defines the voltage in millivolts at the analog pin, when no current is being measured.
The voltage sensor uses the “R1” and “R2” variable to determine the needed resistor ratio for the formula to calculating the input voltage of the sensor. In our Device R1 is 10 kilo-ohms and R2 is 4 kilo-ohms, allowing voltage measurements up to 17.5 volts. Of course, the resistors can always be replaced to achieve higher precision on lower voltages or measure larger voltages, but for the intended use case, this range should be enough.

The temperature sensor uses the steinhart-hart coeficients “c1”, “c2” and “c3” specified in the data sheet of the thermistor to determine the temperature based on the resistivity change of the thermistor.

The actuators, aside from the servo, use the input from the sensor to determine if they should be activated or not. Both the LEDs and the buzzer use the measured voltage, temperature and current to check if the set threshold has been exceeded. The function “onMqttMessage” is used to update the input variables in the system as seen in figure 9.



```
void onMqttMessage(int messageSize) {
  // we received a message, print out the topic and contents
  Serial.println("Received a message with topic '");
  currentTopic = mqttClient.messageTopic();
  Serial.println(currentTopic);
  Serial.print("', length ");
  Serial.print(messageSize);
  Serial.println(" bytes:");
  mqttString = "";

  // use the Stream interface to print the contents
  while (mqttClient.available()) {
    // Serial.print((char)mqttClient.read());
    mqttMessage = (char)mqttClient.read();
    mqttString += (mqttMessage);

  }
  finalvalue = mqttString.toFloat();
  Serial.println(finalvalue);

  if (currentTopic == "Voltage")
  {
    voltage = finalvalue;
  }
  else if (currentTopic == "Current")
  {
    current = finalvalue;
  }
  else if (currentTopic == "Temperature")
  {
    temperature = finalvalue;
  }

  Serial.println();
  Serial.println();
}
```

Figure 9: Function for reading an incoming MQTT message and updating the corresponding variable

The implemented wiring of all the components in the actuator system and sensor system is visualized in figure 11 and 10 respectively.

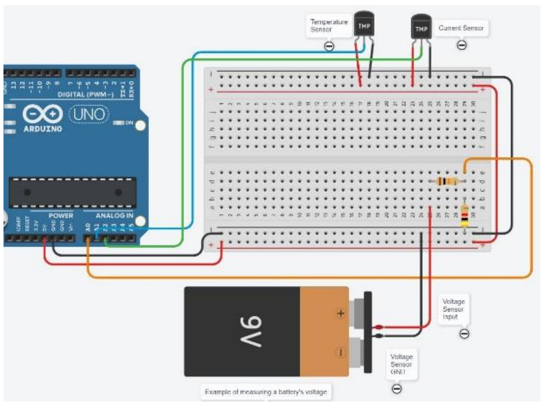


Figure 10: circuit diagram of the sensor system

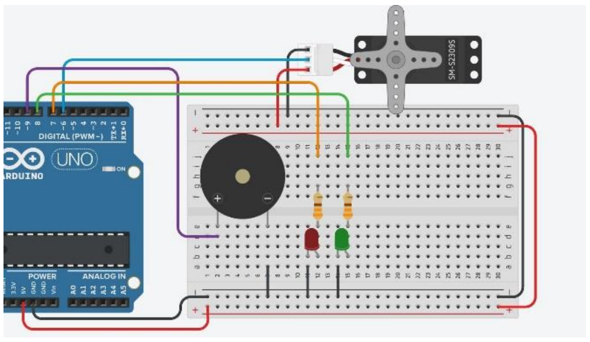


Figure 11: Circuit diagram of the actuator system

The intended use case for our system is in industrial testing environments. When testing electrical systems with high voltage inputs, the intermediate circuits or some driver or control boards with lower operating voltages could need voltage, current and/or temperature measuring. Using our system the results will be immediately displayed to the testing engineer and critical values will trigger an alarm so the engineer can react. In this high voltage application, our system will use a door lock to prevent any person from entering the test environment during testing. If it is not a high voltage test, the door lock can simply be disabled by the user.

7 Use Case

Once the system is assembled as shown in the circuit diagrams of both actuators and sensors, the first step is to turn on the Raspberry Pi by simply connecting it with the proper power supply. It is important that Mosquitto is installed on the device and is set to automatically start after it is booted up. The broker is now ready for operation. Next, connect both arduino's to electricity and wait for a few seconds. Both Arduino's will connect to the Wi-Fi of the Raspberry Pi and to the broker. Once connected, the sensor system will start measuring and send every result to the according topics. Therefore, the actuator system will react according to the set thresholds. If this is not the case, please reset the actuator and/or the sensor system using the reset button on the Arduino to establish the connection anew. To see the exact values of the sensor system one can subscribe to the topics “Current”, “Voltage” and “Temperature” using the “IoT MQTT Panel” app on a smartphone or the “MQTTX” program on computer devices. To connect these applications, the device where the application is used on needs to connect to the Wi-Fi “piSense” first with the password “12345678”. Next, the user has to connect to the broker using the IP address “10.42.0.1” with port number “1883” and the TCP network protocol. Once connected, the user has to create the topics that were mentioned earlier and they will receive all the measured values of each topic as a result, if the connection has succeeded. Additionally, the user can also send a message to the actuator system with the topic “Door” to move the door lock (Servo Motor) back and forth. The message can be “open”, “close”, or simply “door”, which will open and close the lock depending on its last state.

8 Sources/References

Provide the sources on the technologies and algorithms you used in your project (Github).

https://cdn-reichelt.de/documents/datenblatt/A200/ACS712.pdf (Current sensor) [1]
https://www.amazon.de/Stromsensor-Spannungssensor-Hall-Effekt-Stromsensormodul-Terminal/dp/B09F6CT827/ref=sr_1_3_sspa?__mk_de_DE=%C3%85M%C3%85%C5%BD%C3%95%C3%91&crid=23YP1U2ZQ17HA&keywords=current+sensor+arduino&qid=1683640445&sprefix=current+sensor+arduino%2Caps%2C176&sr=8-3-spons&sp_csd=d2lkZ2V0TmFtZT1zcF9hdGY&psc=1 (current sensor amazon) [2]

http://www.hamptonsailer.com/Fubar/37-SENSOR-KIT-TUTORIAL.pdf (Temp sensor) [3]

https://mqttx.app/ [4]

https://play.google.com/store/apps/details?id=snr.lab.iotmqttpanel.prod&hl=de&gl=US&pli=1 [5]