Project Report

Date: 11/30/17

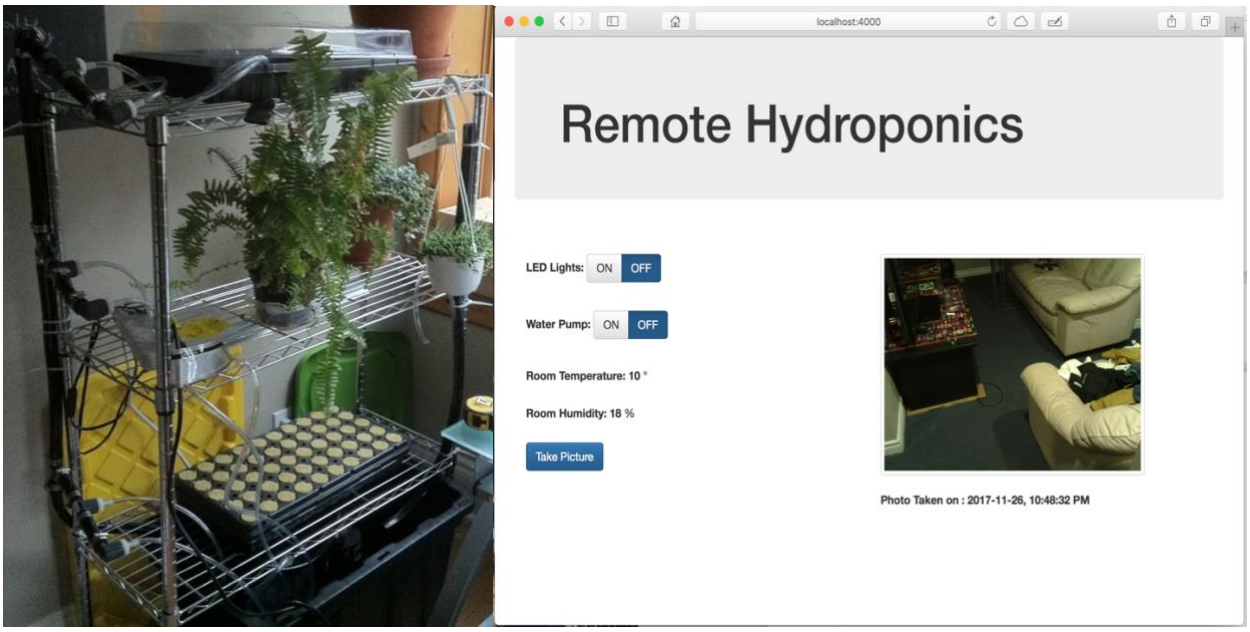
Alex Coghill – 0894734

Nick Major – 0879292

Word Count: 581

Project Description

We set up sensors to remotely monitor and manage a miniature hydroponics farm. Hydroponics is the method of growing plants without soil, using mineral nutrient solutions in a water solvent. Terrestrial plants may be grown with only their roots exposed to the mineral solution, sprayed onto them through a fine mist. We have a couple trays of leafy greens that would be growing.



We measured temperature and ambient humidity, set up a camera to take photos to monitor plant growth. Additionally, we added relays to control the pump and lights for the system. We also created a website to remotely manage the sensors, camera, and relays.

Communication

In our current configuration, we are using an IOT approach for communication. The sensor and relays are connected to the arduino (MRK1000), which is connected to MQTT via WIFI. The MQTT server is hosted remotely on my server and is used to connect the rasberry PI to the arduino. The raspberry pi is used to host a website using node.js to send commands to the arduino and control the pi camera. The front-end of the website (angular.js) communicates with the PI with Websockets. So, when a user toggles one of the lights, a message is sent to the server

using websockets. Then, the server sends a MQTT message to the arduino, which toggles the corresponding relay. Additionally, the arduino is programmed to measure the temperature/humidity sensor every 10s and send a message over MQTT if one of the measurements has changed. Then, the server sends a message over websockets to the front-end with the updated value(s). See the Appendix for all of the code implementing the communication.

MQ Telementry Transport

MQTT was developed at IBM in 1999 as a cost-effective, reliable way to connect monitoring devices used in the oil and gas industries with remote enterprise servers. It is a light-weight publish/subscribe messaging system, that is ideal for constrained networks (low bandwidth, high latency, data limits, and fragile connections). The Publisher and subscriber are autonomous, meaning that neither party know the presence of the other.

Websockets

WebSockets represent a long awaited evolution in client/server web technology. They allow a long-held single TCP socket connection to be established between the client and server which allows for bi-directional, full duplex, messages to be instantly distributed with little overhead resulting in a very low latency connections.

Technologies Used

Sofware

- Node.js
- Angular.js
- Arduino c/c++

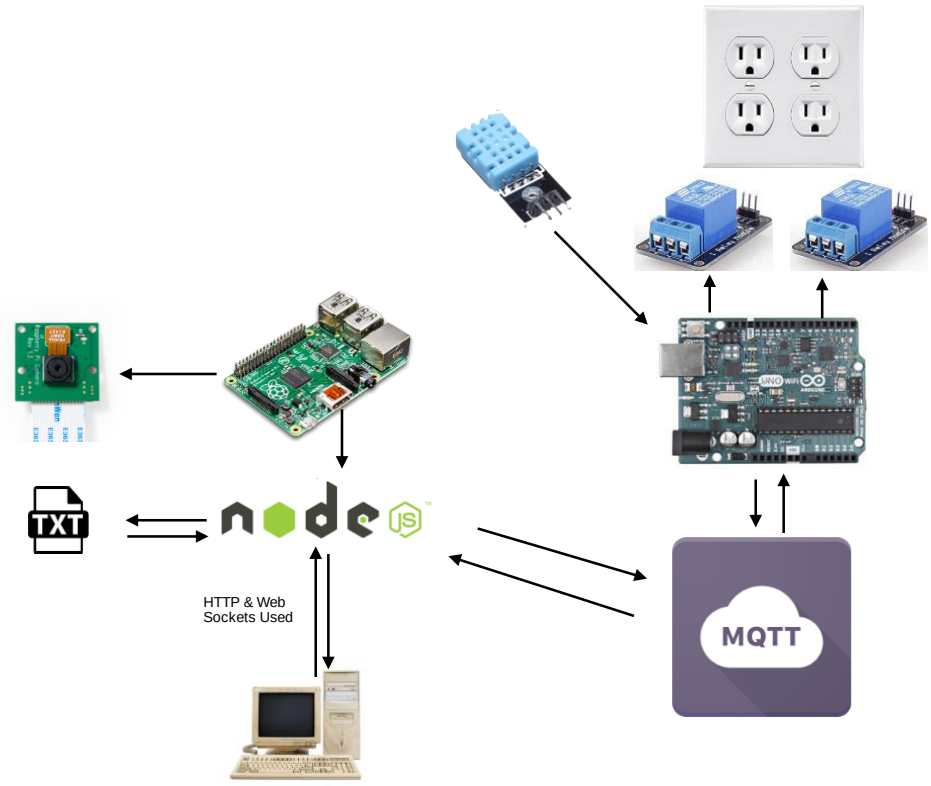
Protocols

- Http (REST)
- Websockets
- MQTT
- WIFI

Hardware

- MKR1000
- PI
- Temperature/humidity Sensor
- PI camera
- Constructed Controllable outlets with 2 5V relays.

Project Diagram



Appendix

Figure 1: Arduino code to send humidity/temperature value to node.js server via MQTT

```
65   if (millis() - lastTimeItHappened >= 10000) {
66     lastTimeItHappened = millis();
67
68     int chk = DHT.read11(DHT11_PIN);
69     Serial.print("Temperature:");
70     Serial.println(DHT.temperature);
71     Serial.print("Humidity: ");
72     Serial.println(DHT.humidity);
73
74     if(DHT.temperature != temp) {
75       temp = DHT.temperature;
76       sprintf(message, "{\"device\":\"temperature\", \"value\":%d}", temp);
77       client.publish("server/", message);
78     }
79
80     if(DHT.humidity != hum) {
81       hum = DHT.humidity;
82       sprintf(message, "{\"device\":\"humidity\", \"value\":%d}", hum);
83       client.publish("server/", message);
84     }
85   }
```

Implemented a timer, that will check the temperature/humidity every 10 seconds. It send a message over MQTT to node.js (that's running on the pi), if either value changes.

Figure 2: Arduino code to receive MQTT messages from node.js server

```
90  void messageReceived(String topic, String payload, char * bytes, unsigned int length) {
91    char str[80];
92    String sensor;
93    String value;
94    Serial.println("Message Received");
95
96    payload.toCharArray(str, 80);
97
98    sensor = strtok(str, ":");
99    value = strtok(NULL, ":");
100
101   if(sensor == "pump") {
102     if(value == "true") {
103       digitalWrite(pumpRelay, LOW);
104     } else if(value == "false") {
105       digitalWrite(pumpRelay, HIGH);
106     }
107   } else if(sensor == "lights") {
108     if(value == "true") {
109       digitalWrite(lightRelay, LOW);
110     } else if(value == "false") {
111       digitalWrite(lightRelay, HIGH);
112     }
113   }
114 }
```

We used an MQTT library for the arduino, so we added our required functionality to a library defined function. We parse and verify the command and send sginals to the correct relay.

Figure 3: Node.js server code running on PI to receive/send MQTT and receive/sent Websocket

```
client.on('message', function (topic, message) {
    var str = decoder.write(message);
    var obj = JSON.parse(str);

    if(obj.device == "temperature" || obj.device == "humidity") {
        io.sockets.emit('sensor',{'topic':String(topic), 'payload':String(message)});
    }

    fs.readFile('data.json', function(err, content) {
        if(err) throw err;
        var parseJson = JSON.parse(content);
        if(obj.device == "temperature") {
            parseJson.temperature = obj.value
        } else if(obj.device == "humidity") {
            parseJson.humidity = obj.value
        }

        fs.writeFile('data.json', JSON.stringify(parseJson), function(err) {
            if(err) throw err;
        });
    });
});

io.sockets.on('connection', function (socket) {
    socket.on('lights', function (data) {
        client.publish('arduino/', data)
    });

    socket.on('pump', function (data) {
        client.publish('arduino/', data)
    });

    socket.on('camera', function (data) {
        io.sockets.emit('camera','start');
        setTimeout(function() {
            io.sockets.emit('camera', 'end');
        }, 6000);

        const exec = require('child_process').exec;
        var yourscript = exec('sh camera.sh',
            (error, stdout, stderr) => {
                console.log(`${stdout}`);
                console.log(`${stderr}`);
                if (error) {
                    console.log(error);
                }
            });
        });
  });
```

Node.js running on the pi, basically does all of the descion making. Connects the front-end of the website to the arduino.

Figure 4: Angular.js Frontend receiving Websocket messages

```
31      socket.on('sensor', function(msg) {
32          var obj = JSON.parse(msg.payload);
33          $('.' + obj.device).html(obj.value);
34      });
35
36      socket.on('camera', function(msg) {
37          console.log(msg);
38          if(msg == "start") {
39              app.cameraDisable = true;
40              showModal();
41          } else if(msg== "end") {
42              hideModal();
43              app.cameraDisable = false;
44              var t = new Date();
45              app.time = t.toLocaleString();
46              $('.img-thumbnail').attr("src", "assets/images/plant.jpeg");
47              var time = {};
48              time.time = t.toLocaleString();
49              Plant.putTime(time);
50          }
51      });
```

Updates temperature/humidity values and image on receiving messages.