**HW #3: MQTT Raspberry Pi Introduction Group 8**

**Team Members:**

1. Stepan Kalinin skalini@ncsu.edu
2. Connor Smith cpsmith6@ncsu.edu
3. Sagar Hirenallur Prasannakumar shirena@ncsu.edu
4. Rishab Gujarathi rgujara@ncsu.edu

|  |  |
| --- | --- |
| **Percent Contribution** | |
| Stepan Kalinin | 25% |
| Connor Smith | 25% |
| Sagar Hirenallur Prasannakumar | 25% |
| Rishab Gujarathi | 25% |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **TASKS** | | Stepan Kalinin | | Connor Smith | | Sagar Hirenallur Prasannakumar | | Rishab Gujarathi | |  |
| MQTT Broker | Code | 100% | |  | |  | |  | |  |
| Debug | 100% | |  | |  | |  | |  |
| Report | 100% | |  | |  | |  | |  |
| Raspberry Pi A | Code | 100% | |  | |  | |  | |  |
| Debug | 100% | |  | |  | |  | |  |
| Report | 100% | |  | |  | |  | |  |
| Raspberry Pi B | Code |  | |  | |  | | 100% | |  |
| Debug |  | |  | |  | | 100% | |  |
| Report |  | |  | |  | | 100% | |  |
| Raspberry Pi C | Code |  | | 100% | |  | |  | |  |
| Debug |  | | 100% | |  | |  | |  |
| Report |  | | 100% | |  | |  | |  |
| Laptop 2 | Code |  | |  | | 100% | |  | |  |
| Debug |  | |  | | 100% | |  | |  |
| Report |  | |  | | 100% | |  | |  |
| Report | | | 25% | | 25% | | 25% | | 25% | |

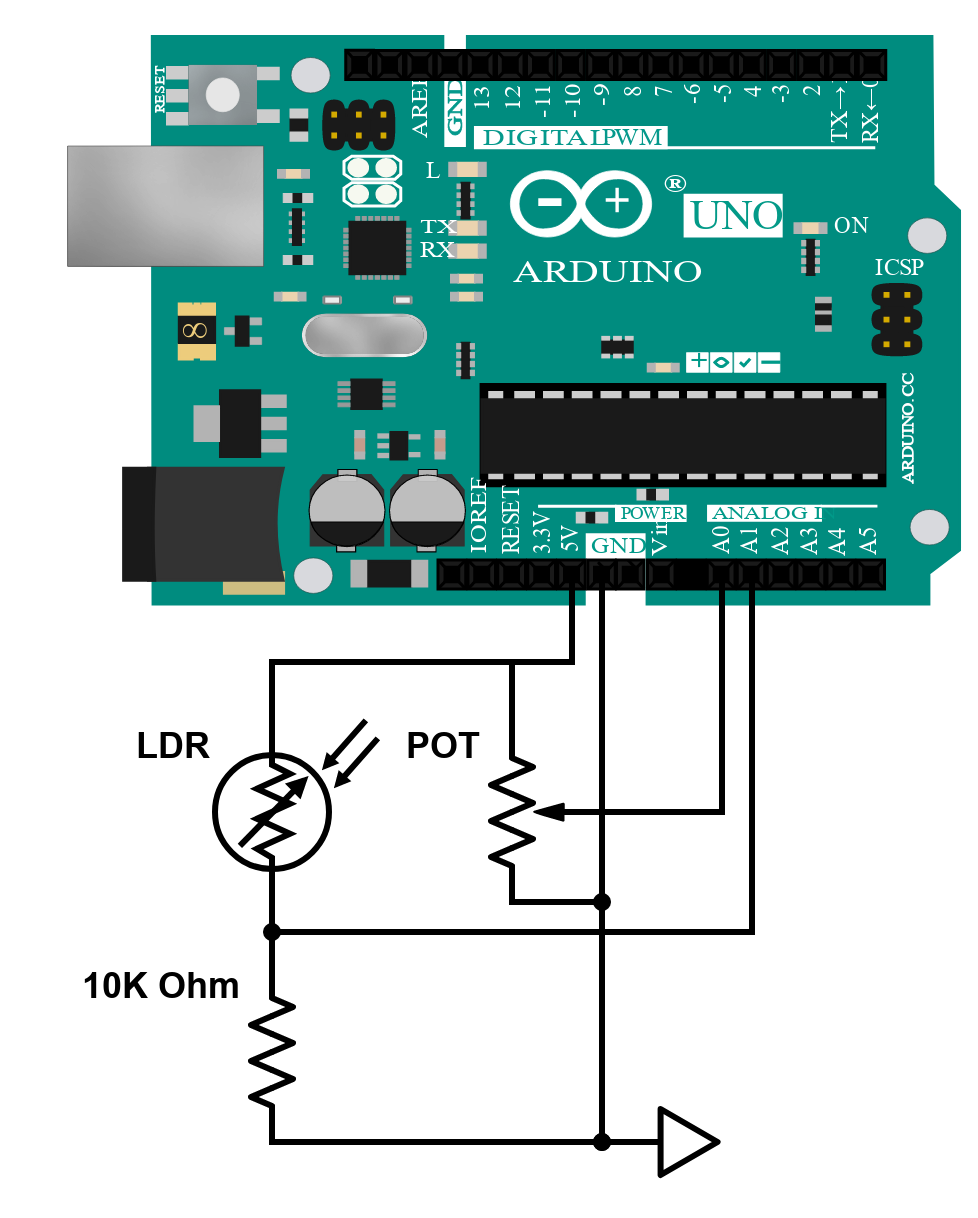
**1. Objective**

For this assignment, we are attempting to have 3 Raspberry Pi devices communicate with an MQTT broker to use the LDR output generated by Raspberry Pi A to illuminate a light on Raspberry Pi B. This is done by Raspberry Pi C receiving LDR and threshold values to publish whether the light should turn on or off for Raspberry Pi B. All the published messages across all Raspberry Pis are logged to a second laptop.

**2. Description**

**2.1 Raspberry Pi A**

**2.1.1 Circuit setup**



Instead of an ADC we used an Arduino to read the analog values. The Arduino is running a Telemetrix

sketch for communication with Raspberry Pi.

**2.1.2 Design decisions**

As specified in the assignment, the sampling rate was set at 10 Hz. The values observed from the potentiometer were 0-1023 (the entire range), and the values observed from LDR were 0-1017. Because of that, no scaling was done and the values were reported raw.

**2.2 Broker**

We chose Mosquitto as our broker for its popularity and ease of setup. The installation instructions can be found in the broker/ directory in the submission.

**2.3 HTTP**

**Development**

Testing the HTTP protocol was done in a fairly similar way to the CoAP protocol. The server is done using Python’s built-in server command to host a simple HTTP server at a given host and port number. Localhost is used for testing on port 80. The client file is ran with option command line arguments for host/port if they are different than localhost:80. A HTTP GET request is sent a specified number of times. A timer is used before and after the GET request is made to track the time required for each GET request. The size of the header and payload is determined by finding the point in the byte string where the headers end and the payload begins. This information allows us to find the size of the header and payload. The Measurement class is also used to return the time, total size, and payload size of each file that is sent. Statistics are then printed to the console.

**Observation**

Examining the results for throughput, we can see that throughput slows considerably for files of a really small and really large file size. HTTP excels with files or medium file size, with throughputs in the 10s of thousands of kbps. For large files, this may be due to network bottleneck constraints. Another possible explanation could be due to the maximum size of each packet, so the data needs to be fragmented more, which causes slower throughput. For medium sized files, we don’t reach the network bandwidth capacity, and the overhead on the packets is comparatively minimal, so we see much higher throughput values. This is also backed up by examining the overheads of the different files. In general, the smaller sized files have significantly more overhead than the larger files.

**3. Comparison**

In looking at throughputs of the 100B file for the four protocols, we can observe that CoAP does a significantly better job than the 3 other protocols. This is possible since CoAP is built in UDP instead of TCP, allowing for significantly faster throughputs. On the flip side, with the 10MB file, CoAP falls far behind the other protocols in throughput. This could be for several reasons. Firstly, since CoAP is built on UDP, there may be issues with packet loss that are exacerbated with larger files that are not present in the other protocols. Second, CoAP is designed to transfer smaller payloads, which is why it is so popular with IoT applications. Fragmentation slows down the transfer process with each fragment also needing its own header. Lastly, bandwidth constraints could also make the previous 2 reasons even worse. When examining the jumps in throughputs between the 10kB file and the 1MB file, there are significantly increases of magnitude in throughput for all the protocols, except for CoAP. This is because the overhead for all the other protocols becomes drastically smaller by comparison as the file size increases. Since CoAP is designed for small payloads, the overhead will stay relatively the same, even as the file size increases because of fragmentation. The overhead decreases for the other 3 protocols because the payloads can transmit more data. Looking at the overhead for the 4 protocols, we can see that CoAP has a much smaller maximum payload size compared to the other protocols. We can see that CoAP reaches its maximum payload size in between the 100B file and the 10kB file because the overhead stays relatively constant after the 100B file, whereas all the other protocols have their overheads decrease by magnitudes of 10 even up to the 10MB file. Our last observation we have made is that MQTT QoS1 is substantially faster than QoS2. This is because QoS1 is sending only 1 acknowledgement during data transmission, whereas QoS2 sends multiple acknowledgements to ensure the safety of the data transmission, at the cost of speed.