WESTERN NEW ENGLAND UNIVERSITY WNE

ELECTRICAL AND COMPUTER ENGINEERING

To: Prof. Qouneh

From: Ethan Griswold and Jeremy Burke **Section Number:** CPE-323-02

Date: May 10th, 2025

Subject: Honors Project – LCD and Mosquitto Broker on BeagleBone Black

Executive Summary
This project consists of two parts. The first part is modifying Derek Molloy's provided circuit and code to use an LCD screen with the BeagleBone Black. The second part of this project was to set up the Mosquitto MQTT broker on the BeagleBone Black and write sample programs to show its operation.

All changes and steps taken have been documented to allow for reproduction of our results. In addition, all code, circuit diagrams, and referenced articles have been attached separately in a .zip file. They can also be found in the following GitHub repository: https://github.com/egriz785/HonorsProject2025

Part 1 – LCD screen

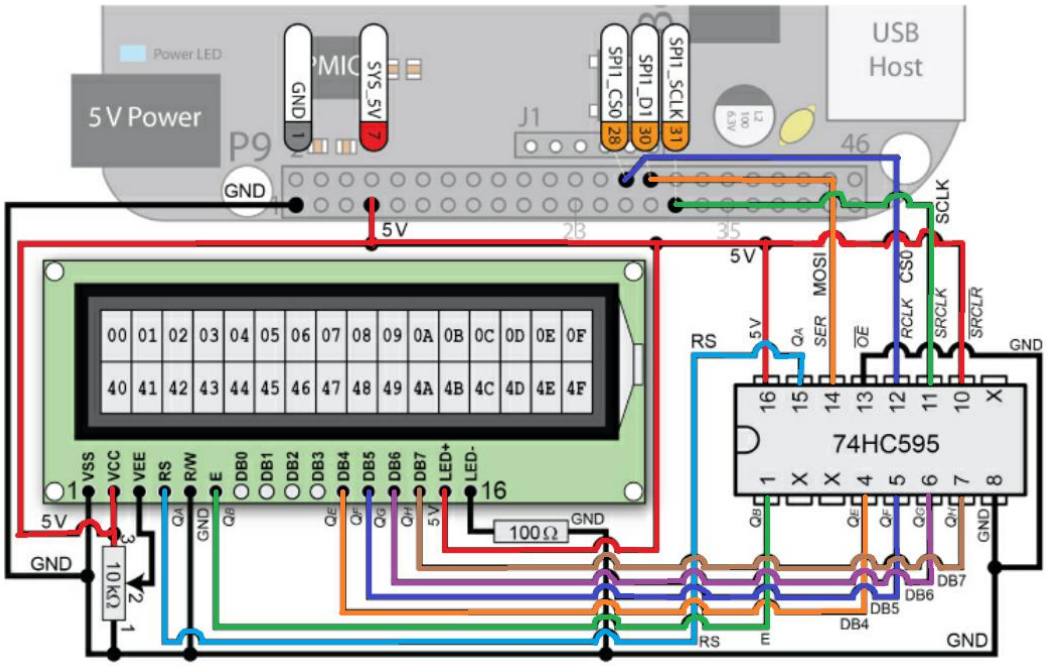

Figure 1, Derek Molloy's schematic for the LCD screen using SPI1.0 (with colors added)

The above schematic can be found in Derek Molloy's book *Exploring BeagleBone*. To make the wiring easier to follow, we colorized the wires. To communicate with the LCD screen, the BeagleBone uses the SPI protocol. Unfortunately, Molloy chose to use the SPI1.0 pins on the BeagleBone, which by default cannot be used since those pins also control the HDMI output. Instead of disabling the HDMI output and reconfiguring all of the pins, we chose to move to the SPI0.0 pins, which required no setup other than using the config-pin tool.

Below are the changes in wiring required:

SPI1 pins	SPI0 pins
SPI1_CS0 (p9.28)	SPI0_CS0 (p9.17)
SPI1_D1 (p9.30)	SPI0_D1 (p9.18)
SPI1_SCLK (p9.31)	SPI0_SCLK (p9.22)

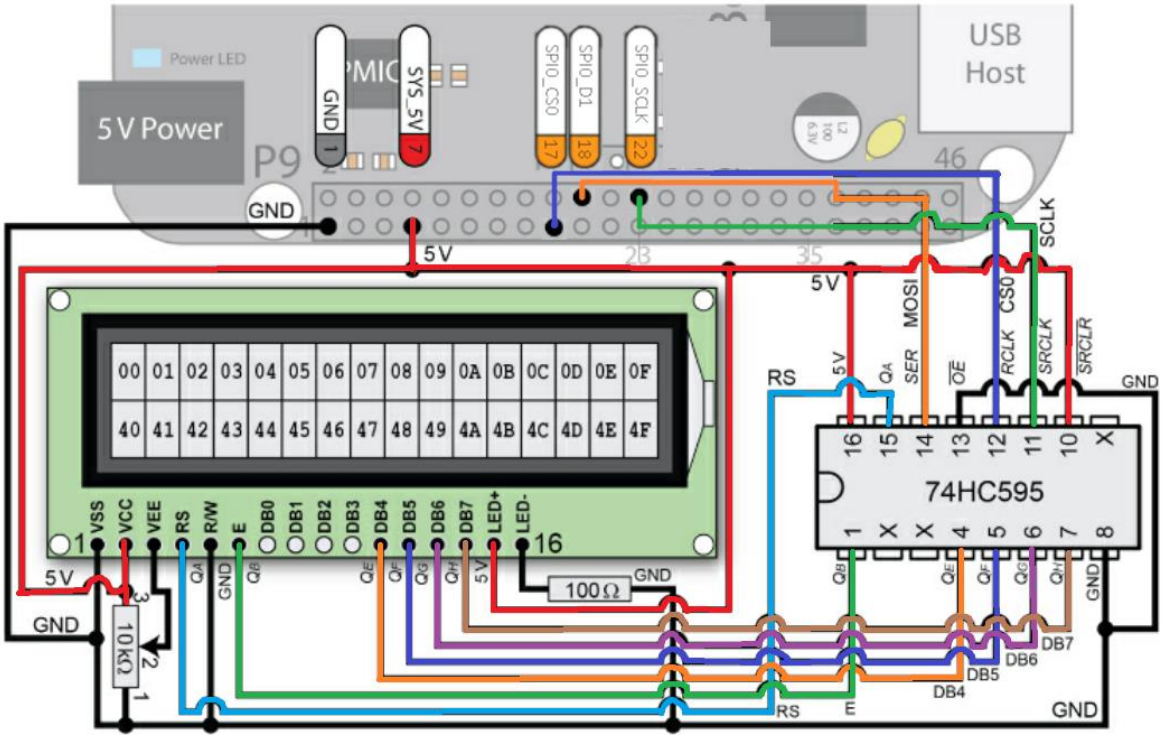

Figure 2, updated schematic for SPI0.0

Molloy also provided classes and sample code to write to the LCD screen, which can be found in the Exploring BeagleBone GitHub repository. The LCD files can be found at:

exploringBB/chp09/LCDcharacter

We had issues compiling these files, since he used static libraries and shared objects. To get around this, we moved all of the relevant class files into the working directory and changed the build command. The class files can be found at:

exploringBB/library/bus:

- BusDevice.h
- BusDevice.cpp
- SPIDevice.h
- SPIDevice.cpp

exploringBB/library/display:

- LCDCharacterDisplay.h
- LCDCharacterDisplay.cpp

We chose to move them to the working directory, but to keep them inside of their bus and display directories respectively, which is reflected in the updated build command found below:

Old build command:

```
g++ LCDApp.cpp ../../library/libEBBLibrary.so -o LCDApp -I "../../library"
```

New build command:

g++ LCDApp.cpp display/LCDCharacterDisplay.cpp bus/SPIDevice.cpp bus/BusDevice.cpp -o LCDApp

In addition, LCDApp.cpp creates an instance of an SPI device, provided an SPI port number. The original file uses SPI1.0, so the file reads:

SPIDevice *busDevice = new SPIDevice(1,0)

But for this program to output to SPI0.0, it needs to be updated to:

SPIDevice *busDevice = new SPIDevice(0,0)

To show off the functionality of the LCD screen, we also wrote a program that takes in two command line arguments and prints them to the LCD screen. The first argument prints on the top row, and the second on the bottom row. Then, it finds which string is shorter, pads it with spaces equally on both sides so the two strings are the same length, and scrolls it across the screen from left to right. This is the LCDScroll program found under the Part1_LCD folder.

Included with these programs are a build file, the class files, and a bash file called "set_pins" that runs the config-pin tool to set all of the SPI pins to their respective functions.

Part 2 – Mosquitto MQTT Broker

The Mosquitto MQTT broker is an open source software that is simple to set up on a Linux environment. When doing some research into Mosquitto, we found two articles that were very helpful and gave clear instructions to install and configure the broker.

To install the Mosquitto broker, the article recommends adding Mosquitto to the Debian repository. We found this was an unnecessary step but have included the commands in case future operating systems no longer support this.

```
wget http://repo.mosquitto.org/debian/mosquitto-repo.gpg.key
sudo apt-key add mosquitto-repo.gpg.key
cd /etc/apt/sources.list.d/
sudo wget http://repo.mosquitto.org/debian/mosquitto-jessie.list
sudo apt-get update
```

Next was to install the broker itself:

```
sudo apt-get install mosquitto
```

In addition, the guide recommended installing the Mosquitto client, which allows you to use the command line for publishing and subscribing to topics directly. We found this particularly useful both during the initial setup of the broker to verify that it was working, as well as testing our C++ programs.

```
sudo apt-get install mosquitto-clients
```

Once installed, the Mosquitto service begins running immediately. The following command can be inputted to the terminal to check the status:

```
sudo service mosquitto status
```

To stop, start, or restart the broker, the following commands can be used:

```
sudo service mosquitto start
sudo service mosquitto stop
sudo service mosquitto restart
```

For testing the broker, you can use the command line client to subscribe and publish to a topic. One test that we did early on was to open two terminals and subscribe to a topic “testtopic”, and then on the other terminal publish a message. This can be done using the following commands:

To subscribe:

```
mosquitto_sub -t "testtopic"
```

To publish:

```
mosquitto_pub -t "testtopic" -m "Testing"
```

Note, these will only work if you installed the client.

By using the --help flag on either the pub or sub command, it will give a list of all other flags and what they do. The important ones are:

-h : The IP of the broker. Defaults to localhost
-p : The port that you wish to send the message on. By default, this is 1883
-t : The topic, for example –t “testtopic” or - “temperature”
-m : The message you wish to publish. This is not used when subscribing
-u : Username (AUTHMETHOD in the paho templates)
-P : Password (AUTHTOKEN in the paho templates)
-i : ID, will default to mosquitto_pub_PROCESSID (CLIENTID in the paho templates)

In Molloy’s paho templates, the broker IP and port are combined as the ADDRESS, defined at the top as (broker IP):(port), for example 127.0.0.1:1883.

If you wish to connect a second BeagleBone to your broker, you will need the broker’s IP address. You can find the IP of the broker by running the command on the broker:

```
ifconfig
```

You do not need to know the IP of the second BeagleBone, only the IP address of the broker to connect to it. Typically, the two devices do need to be on the same network to connect, at least for the scope of this project.

By default, there is no username or password required to publish or subscribe on the Mosquitto broker, and you are not required to set one up to use the paho example templates or the client commands. If you do wish to set up any usernames and passwords, the process is the following:

Stop the broker:

```
sudo stop mosquitto
```

Create a new password file. This creates a new file in /etc/mosquitto/passwd that contains the username and encrypted password.

```
sudo mosquitto_passwd -c /etc/mosquitto/passwd (YOUR USERNAME)
```

It will then have you enter and reenter a password. Mosquitto only allows for one username and password, so running this command an additional time will overwrite the previous username and password. To delete the username and password, run the following command:

```
sudo mosquitto_passwd -D /etc/mosquitto/passwd (YOUR USERNAME)
```

Now, you must edit the configuration file for the broker, so it knows where the username and password are kept.

```
sudo vim /etc/mosquitto/mosquitto.conf
```

Add the following line to the end of the file:

```
password_file /etc/mosquitto/passwd
```

If you would like to require a username and password, also add:

```
allow_anonymous false
```

If you choose to add the second “allow_anonymous false” line, attempting to connect to the broker without using the correct username or password will result in an error.

To start the broker and use the new config file, run the following:

```
mosquitto -c /etc/mosquitto/mosquitto.conf
```

This is only required once. Every time the broker starts from now on, it will use this configuration file.

Derek Molloy also provided code using the paho library for C++, although he used it to connect to AdafruitIO. These programs can all be configured to connect to the new Mosquitto broker by just changing the definitions at the top.

For example, if you are using Molloy's publish.cpp and subscribe.cpp examples, connecting to the broker using the following parameters:

IP address	192.168.69.100
Port	1883
Topic	temperature
ID	BeagleBone1
Username	UserName
Password	PassWord

Would look like:

```
#define ADDRESS     "192.168.69.100:1883"
#define CLIENTID    "BeagleBone1"
#define TOPIC       "temperature"
#define AUTHMETHOD  "UserName"
#define AUTHTOKEN   "PassWord"
```

in the definition section of both publish.cpp and subscribe.cpp.

A set of example programs have also been provided, under the MQTT_LCD folder. The publish program takes two command line arguments, which are published over MQTT to the topic "lcd”. Then, the subscribe program receives these values and prints them to an LCD screen, using a modified program from Part 1 of this project. The examples have the IP of the broker defined as localhost (127.0.0.1), meaning they will run as expected on the broker, but if you wish to run them on different BeagleBones, the IP must be updated as stated earlier.