# M2M communications on Atmel AVR IoT Boards using MQTT

EE2016 Microprocessors

Sanjeev Subrahmaniyan S B

EE23B102

**Abstract**

Internet of Things(IoT) is the buzzword in state of the art electronics and finds specific importance in embedded systems. An internet of things is a network where devices are connected together in a local network and can communicate between each other over communication protocols. In this experiment, an Internet of Things is implemented on AVR IoT boards using the MQTT protocol(Mosquito Queueing and Transport Telemetery), where data is transmit over WiFi moderated through a mosquitto server. The experiment involves sending messages between handheld devices(mobile phones) using the myMQTT application, and AVR IoT boards. Data such as simple text messages, and luminosity measurements from the light sensor on the board are transmit through the network and displayed on an LCD display. Key insights are gained on the functionality of ADC channels in microcontroller and operation of LCD displays in serial mode. The project is developmed using the MPLAB X IDE.

## 1 Introduction

Mobile to mobile(M2M) communications find a lot of practical applications and serve as one of the leading market for embedded systems. The objective is to broadly be able to send and receive data wirelessly over short ranges with high security. The atmel AVR IoT WG boards are designed specifically for prototyping and modelling such IoT devices and are used in this experiment to perform the communication operations.

Broadly, all the devices on this IoT are connected to a central server - the mosquitto server - over WiFi. The mosquitto server acts are a broker over the network, sending, receiving and rerouting packets between devices. In this experiment, a laptop is configured to be used as the broker, with all the devices connected to it through a mobile hotspot configured on the laptop. These networks together constitute a secure network which can be used to control one device from another and establish communication.

## 2 Objectives

While the experiment's objective is to gain familiarity with IoT communications and implementation, it is divided into the following tasks which were performed in the laboratory:

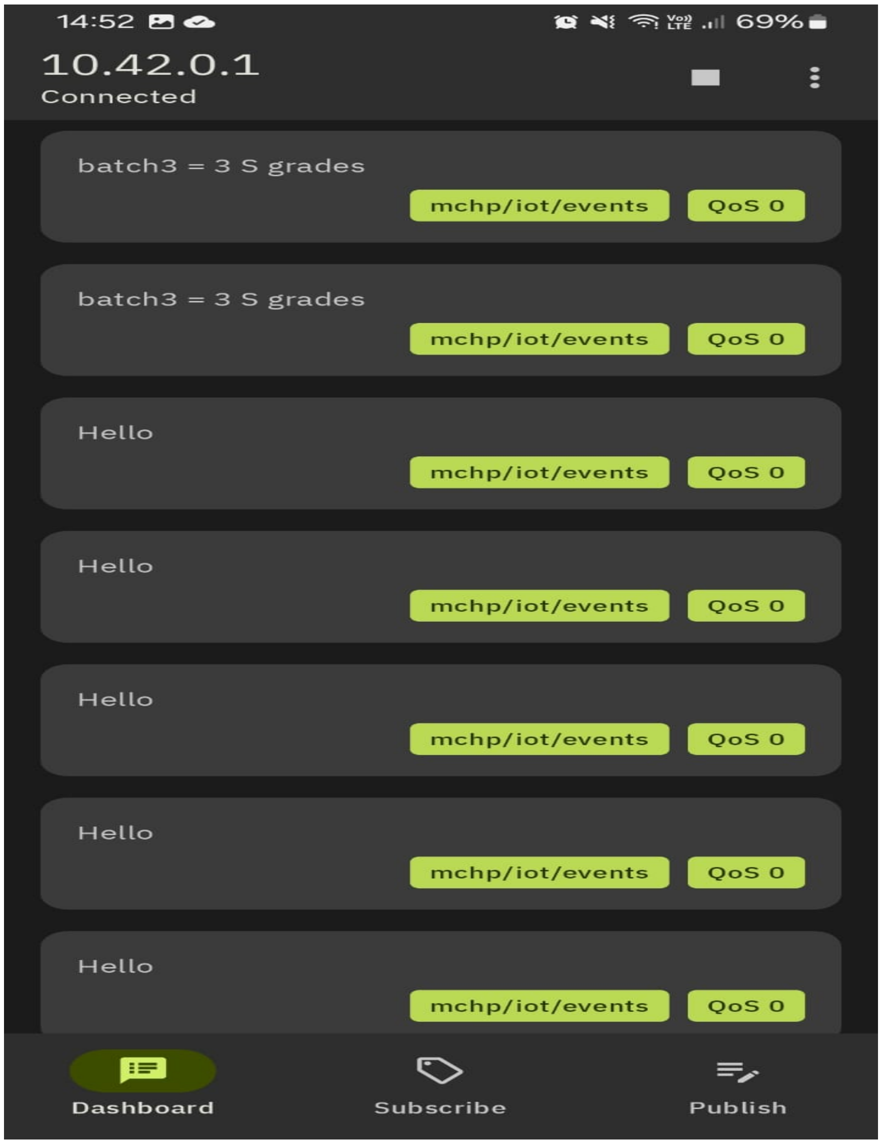
1. Send messages between two mobile phones connected with the myMQTT application
2. Configure two AVR IoT boards are subscriber and publisher, send a short message. from the publisher and blink an LED when the message is received by the subscriber. The subscriber simultaneously publishes messages on an unrelated topic.
3. Send text messages from the publisher, receive it on the subscriber and print the message on an LCD connected to it.
4. Read luminosity values through the light sensor on the publisher mode, use an ADC channel to convert the value to digital and relay it to the subscriber. Display this on the LCD screen.

## 3 Programs and Observations

The experiment was well structured with example programs provided for all subparts, many of which required filling in missing parts to complete the function. The programs in this report are only the ones that were fixed/added by us and the ones provided on moodle are not added to ensure conciseness of the report.

### 3.1 M2M communication using myMQTT on mobile phones

The procedure for this part of the experiment is as given in the handout. All that was required to do was to install the applications on both the mobile phones, setup the mosquitto broker using the given configurations and subscribe and publish to the same topics. The results could be seen on the phone itself. A random message has been published to a random topic as can be seen. There was no program required to be written for this part of the experiment.


14:52

10.42.0.1
Connected

batch3 = 3 S grades
mchp/iot/events QoS 0

batch3 = 3 S grades
mchp/iot/events QoS 0

Hello
mchp/iot/events QoS 0

Hello
mchp/iot/events QoS 0

Hello
mchp/iot/events QoS 0

Hello
mchp/iot/events QoS 0

Hello
mchp/iot/events QoS 0

Figure 1: Screenshots of the implementation on mobile phone

### 3.2 Blinking LED on receiver when message is published by another board

The procedure for this part of the experiment is also the same as that given in the handout. Broadly the steps performed are:

1. Configure the mosquitto broker on laptop and note the IPv4 address of the laptop
2. Create a project for publishing the data on MPLAB X IDE. The project is configured by adding the MQTT(for IoT communication) and WINC(for WiFi access) and setting

them to the parameters are required. The pinouts are configured after the clock settings are made and the program is generated.

3. The main.c and mqtt_example.c files are modified to perform the process of sending a message on the topic 'b3/hello'. This topic was chosen by us to ensure no mixup with other teams at the same time.
4. Similar to the publisher board, another board is configured and the code is generated with identical settings. The only difference is that the new board subscribes to the topic 'b3/hello' and publishes to 'b3/yolo' which is a random topic. This will enable this board to receive the messages from the first board.
5. The codes are generated using the tool on the IDE and modified to serve our function. The LEDs on pins D0, D1, D2, D3 are set to blink everytime a message is received on the subscriber board.

The LEDs on the receiver board blinking can be seen on the photos taken in the lab:

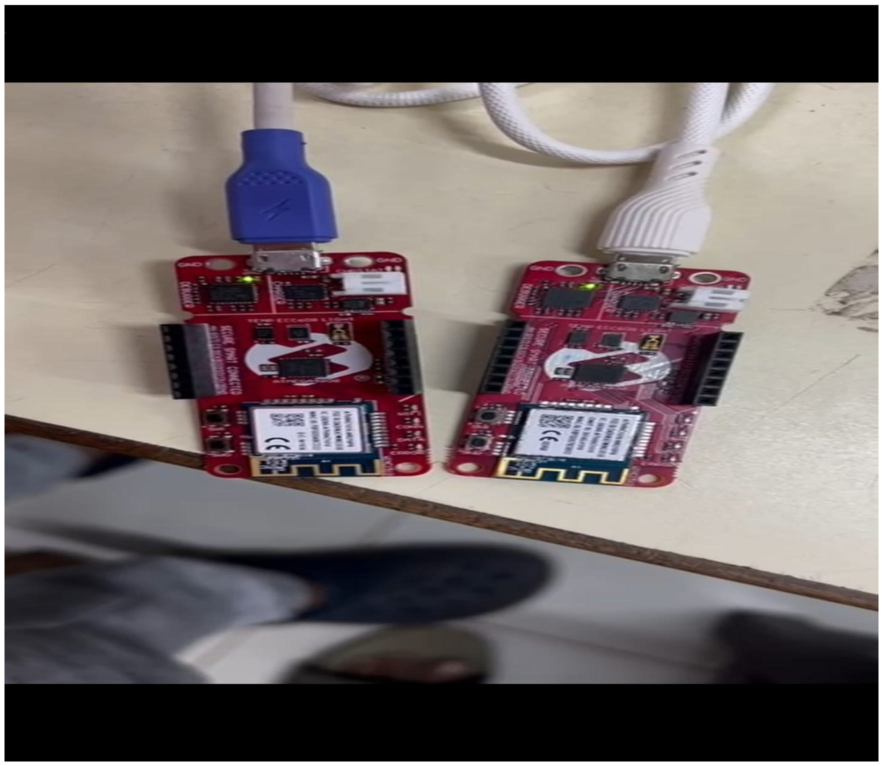


Figure 2: LED blinking on receiver board

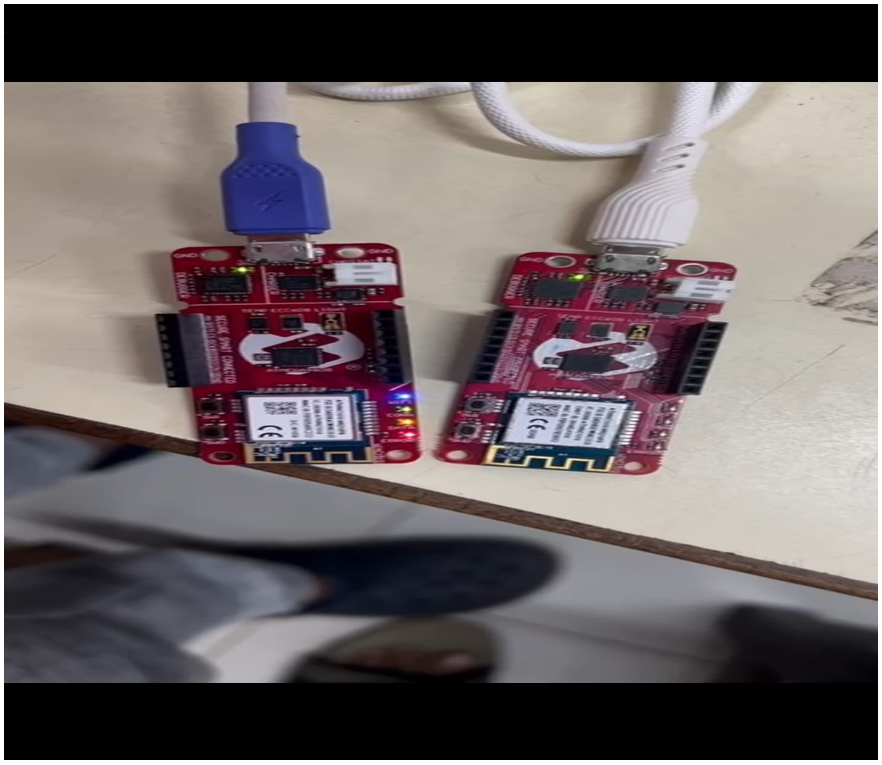


Figure 3: LED blinking on receiver board

### 3.3 Sending Luminosity Data from sensor and displaying on LCD screen

For this part of the experiment, it was required that we read the luminosity values from the light sensor on one board, use the ADC channel to convert it into digital data, transmit it to the other board and display the results on the LCD screen. The configuration of the board for this experiment is same as that for the LED blinking, and all other necessary pin configurations are set in the code itself. The programs for the parts of this experiment follow:

#### 3.3.1 Reading and sending luminosity data

The program is divided into parts for the main program and that for the MQTT. The main program follows, where the luminosity value is read and converted into digital value using the analog to digital converter in the ATMega4808 microcontroller:

```
#include "mcc_generated_files/mcc.h"
#include <avr/io.h>
#include <math.h>
#include <stdio.h>
#include <util/delay.h>
```

```c

void ADC_init(void) {
    ADC0.CTRLC = ADC_REFSEL_VDDREF_gc; // Set the reference voltage (AVcc)
    ADC0.CTRLA = ADC_ENABLE_bm | ADC_RESSEL_10BIT_gc; // Enable the ADC
    ADC0.MUXPOS = ADC_MUXPOS_AIN5_gc; // Select the ADC channel (AIN5 on PD5)
}

uint16_t ADC_read(void) {
    ADC0.COMMAND = ADC_STCONV_bm; // Start conversion
    while (!(ADC0.INTFLAGS & ADC_RESRDY_bm)); // Wait for conversion to complete
    ADC0.INTFLAGS = ADC_RESRDY_bm; // Clear the interrupt flag
    return ADC0.RES; // Return the result
}

float ADC_to_voltage(uint16_t adc_value, float vref) {
    return (adc_value * vref) / 1024; /* 10 bit ADC */
}

float voltage_to_illuminance(float voltage) {
    float a = 20.0 / exp(0.1 * 2.485); // Derived a
    float b = 2.485; // Derived b from the two points

    return a * exp(b * voltage);
}



int main(void) {
    SYSTEM_Initialize();
    app_mqttExampleInit();
    ADC_init();

    char hello_message[] = "Hello World!";
    char lux_message[20];

    while (1) {
        uint16_t adc_value = ADC_read();
        float voltage = ADC_to_voltage(adc_value, 3.3);
        float lux = voltage_to_illuminance(voltage);

        // Format the lux message to send via MQTT
        sprintf(lux_message, "Lux: %.2f", lux);

        // Send the lux value message
        uint8_t lux_mqtt_message[20];
        for (int i = 0; i < sizeof(lux_message); i++) {
            lux_mqtt_message[i] = (uint8_t)lux_message[i];
        }
        app_mqttScheduler(lux_mqtt_message);

    }
}
```

And similar to the mqtt_example.c generated, the mqtt publisher code is written and was uploaded on moodle. That code was used for this purpose.

### 3.3.2 Receving and displaying sensor data

This part of the task involved configuring the receiver board to read the data and route it to the LCD board that was interfaced. The interfacing follows the pin mappings that are found in the datasheet of the Hitachi 16x2 LCD module. The main program for this part is:

```
#include "mcc_generated_files/mcc.h"
#include <avr/io.h>
#include "mcc_generated_files/examples/mqtt_example.h"
#define LCD_PORT_A PORTA
#define LCD_DDR_A  PORTA.DIR
#define LCD_PORT_C PORTC
#define LCD_DDR_C  PORTC.DIR
#define LCD_PORT_D PORTD
#define LCD_DDR_D  PORTD.DIR
#define RS PIN2_bm
#define E  PIN3_bm
#define D4 PIN0_bm
#define D5 PIN1_bm
#define D6 PIN6_bm
#define D7 PIN4_bm

int main(void) {

    // Initialize LCD
    SYSTEM_Initialize();
    app_mqttExampleInit();

    LCD_DDR_A |= RS | E;
```

```c
    LCD_DDR_C |= D4 | D5;
    LCD_DDR_D |= D6 | D7;
    lcd_init();

     // Initialize MQTT client
    while (1) {

        app_mqttScheduler();    // Run MQTT client scheduler
    }
    return 0;
}
```

Similar to the led blink receiver code, the mqtt_example.c code is modified to perform the required operations. The part of the code that was required to be filled by me is attached(the rest is left out to keep the report short):

```c
void lcd_send(unsigned char value, unsigned char mode) {
    // Set RS for command (0) or data (1)
    if (mode == 0) {
        LCD_PORT_A.OUTCLR = RS; // command
    } else {
        LCD_PORT_A.OUTSET = RS; // data
    }

    // Send higher nibble
    LCD_PORT_C.OUTCLR = D4 | D5;
    LCD_PORT_D.OUTCLR = D6 | D7;

    LCD_PORT_C.OUTSET = ((value & 0x10) >> 4) * D4 | ((value &
       0x20) >> 5) * D5;
    LCD_PORT_D.OUTSET = ((value & 0x40) >> 6) * D6 | ((value &
       0x80) >> 7) * D7;

    LCD_PORT_A.OUTSET = E; // This tells the LCD that data is
       ready to be read from the data pins.
    _delay_us(1);
    LCD_PORT_A.OUTCLR = E; // This tells the LCD to latch the
       data.
    _delay_us(200);

    // Send lower nibble
    LCD_PORT_C.OUTCLR = D4 | D5;
    LCD_PORT_D.OUTCLR = D6 | D7;

    LCD_PORT_C.OUTSET = ((value & 0x01) >> 0) * D4 | ((value &
        0x02) >> 1) * D5;
    LCD_PORT_D.OUTSET = ((value & 0x04) >> 2) * D6 | ((value &
        0x08) >> 3) * D7;

    LCD_PORT_A.OUTSET = E;
    _delay_us(1);
    LCD_PORT_A.OUTCLR = E;
    _delay_ms(2);
}

void lcd_init(void) {
    _delay_ms(20); // Wait for LCD to power up

    lcd_send(0x02, 0); // Initialize LCD in 4-bit mode
    lcd_send(0x29, 0); // 2 lines, 5x8 matrix
    lcd_send(0x0c, 0); // Display on, cursor off
    lcd_send(0x06, 0); // Increment cursor
    lcd_send(0x01, 0); // Clear display
    _delay_ms(2); // Wait for the LCD to process the clear
        command
}

void lcd_string(char *str) {
    while (*str) {
        lcd_send(*str++, 1);
    }
}

void lcd_clear(void) {
    lcd_send(0x01, 0); // Clear display command
    _delay_ms(2);      // Wait for the LCD to process the clear
         command
}
void lcd_set_cursor(uint8_t col, uint8_t row) {
    if (row == 0) {
        lcd_send((col & 0x0F) | 0x80, 0);
    } else {
        lcd_send((col & 0x0F) | 0xc0, 0);
    }
}

void MQTT_publish_handler_cb(uint8_t *topic, uint8_t *payload)
   {
    printf("Received MQTT Message - Topic: %s, Payload: %s\n",
        topic, payload);
    equalize_arrays(last_received_topic, topic);
    equalize_arrays(last_received_message, payload);

    // Store the received message
    strcpy(lcd_payload, (char*)last_received_message);
     // Clear the LCD and display the new payload value
    lcd_send(0x01, 0);
    lcd_set_cursor(3,0);
    lcd_string(lcd_payload);


    // Blink the External LED on PD4
    PORTD.OUTSET = PIN4_bm;
    _delay_ms(500);
    PORTD.OUTCLR = PIN4_bm;
    _delay_ms(500);

    //printf("Received Message: %s\n", payload);
    PORTD.OUT = 0X1F;
    _delay_ms(350);
    PORTD.OUT = 0; //turn second LED on (BLUE)
    _delay_ms(500);
    PORTD.OUT = 0X07;
    _delay_ms(350);
    PORTD.OUT = 0; //turn second LED on (Green)
    _delay_ms(500);
       PORTD.OUT = 0X13;
    _delay_ms(350);
    PORTD.OUT = 0; //turn second LED on (YELLOW)
    _delay_ms(500);
        PORTD.OUT = 0X01;
    _delay_ms(350);
    PORTD.OUT = 0; //turn second LED on (RED)
    _delay_ms(500);
       PORTD.OUT = 0X00;
    _delay_ms(350);
    PORTD.OUT = 0; //turn second LED on (Green)
    _delay_ms(500);
```

The setup was tested for the light sensor data by obstructing the sensor and flashing a flashlight on it. The results can very clearly be seen in the images that follow:

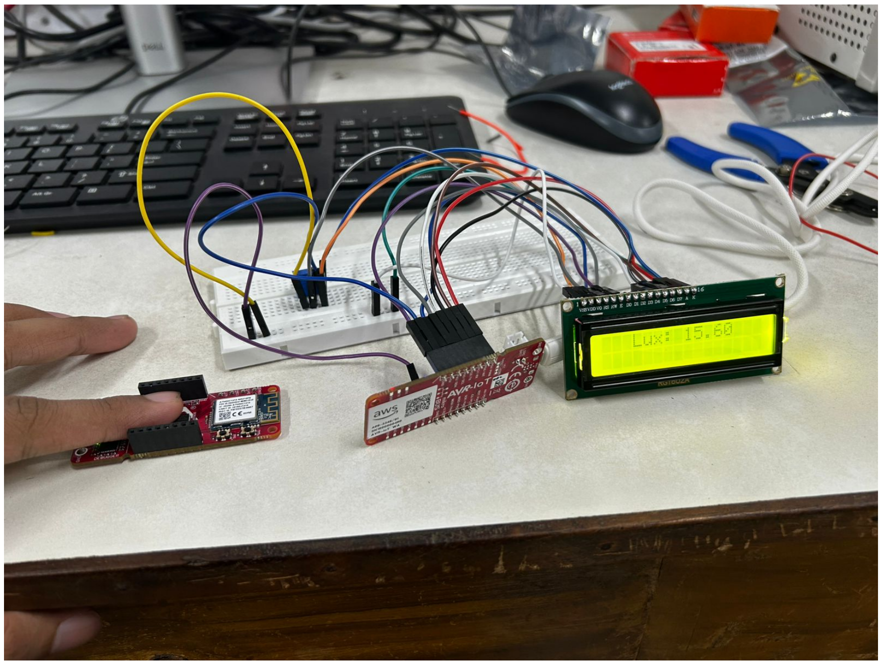


Figure 4: Luminosity value captured with low lighting

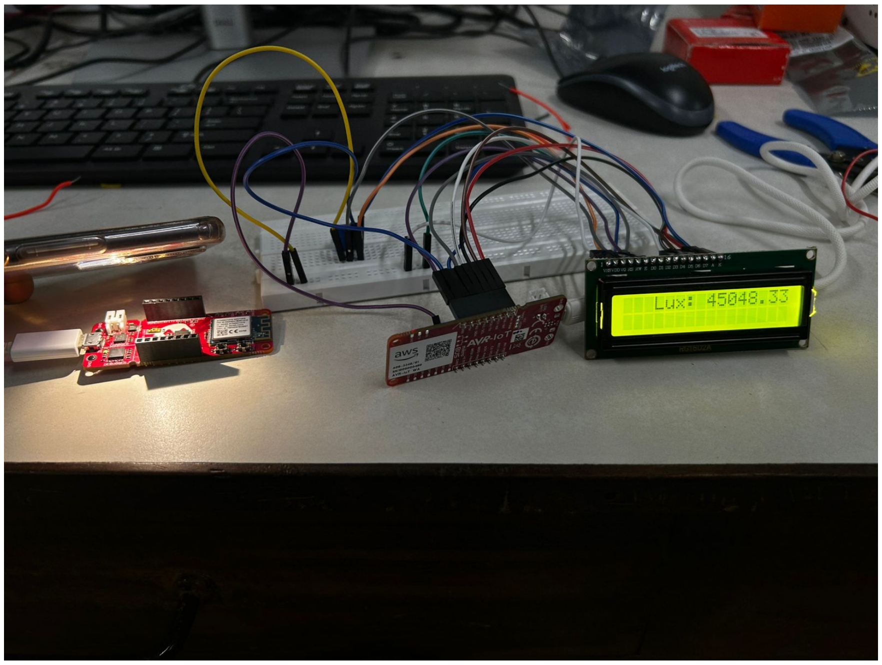


Figure 5: Luminosity data captured with bright lighting

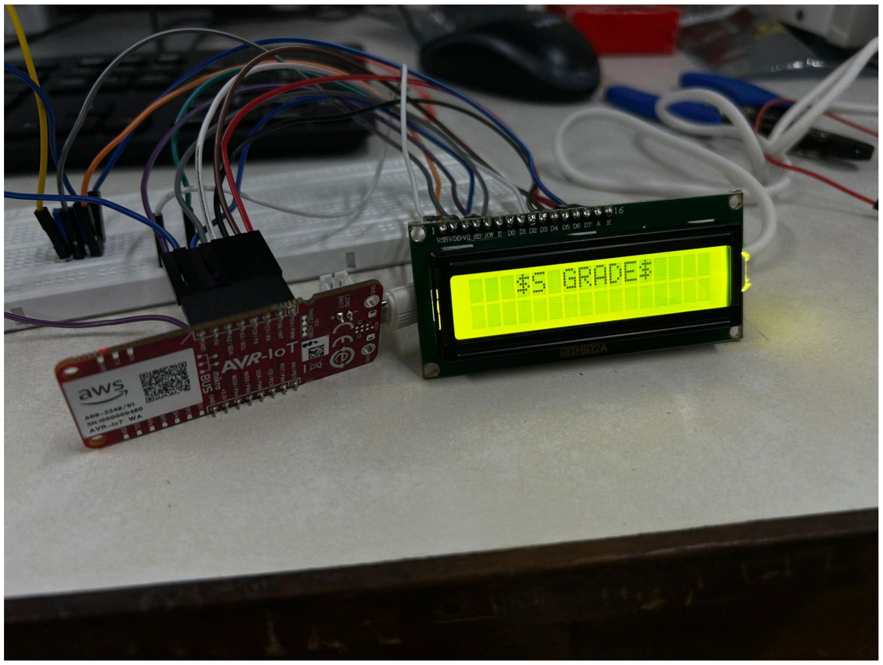


Figure 6: Text message sent from mobile phone being displayed on the LCD

## 4 Conclusion

In conclusion, the implementation of IoT on AVR IoT boards demonstrates the potential of embedded systems in creating smart, connected devices. Through this project, we successfully integrated sensors such as light sensors and multiple mobile devices which communicate through the mosquitto broker in real time. The experience provided valuable insight into the practical challenges of IoT development, such as configuring hardware and writing protocol compliant programs. It also helped me understand the functioning of an ADC channel and interfacing an LCD board with a microcontroller. Altogether, the experiment was very helpful to gain experience on IoT development with MPLAB X and AVR IoT boards and interfacing them with relevant sensors.