ECE 445L Lab 4E Internet of Things

MQTT, Periodic Interrupts, UART, Alarm clock, LCD

This laboratory assignment accompanies the book, [*Embedded Systems: Real-Time Interfacing to ARM Cortex M Microcontrollers, ISBN-13: 978-1463590154*](https://www.amazon.com/Embedded-Systems-Real-Time-Interfacing-Microcontrollers/dp/1463590156), by Jonathan W. Valvano, copyright © 2024.

# Table of Contents

[Table of Contents 1](#_Toc189931327)

[Team Size 2](#_Toc189931328)

[Goals 2](#_Toc189931329)

[Review 2](#_Toc189931330)

[Starter Files 2](#_Toc189931331)

[Required Hardware 2](#_Toc189931332)

[Lab Overview 3](#_Toc189931333)

[Preparation 4](#_Toc189931334)

[Procedure 5](#_Toc189931335)

[Deliverable 1 8](#_Toc189931336)

[Deliverable 2 8](#_Toc189931337)

[Deliverable 3 8](#_Toc189931338)

[Deliverable 4 9](#_Toc189931339)

[Deliverable 5 9](#_Toc189931340)

[Deliverable 6 9](#_Toc189931341)

[Lab Checkout 9](#_Toc189931342)

[Lab Report 9](#_Toc189931343)

[Hints 9](#_Toc189931344)

# Team Size

The team size for this lab is **2**.

# Goals

* Program an ESP8266 01S (ESP8266) to act as a UART-MQTT bridge using the Arduino IDE.
* Interface the ESP8266 with the TM4C using UART.
* Implement a “smart object” that connects to a web application using a MQTT broker.
* Remotely control the clock developed in Lab 3 via the MQTT publish/subscribe interface.

# Review

* Valvano Section 11.4 on IOT
* Chapter 16 of the eBook: [Internet of Things](http://users.ece.utexas.edu/~valvano/Volume1/E-Book/C16_InternetOfThings.htm)
* [Arduino Basics](https://docs.arduino.cc/software/ide-v1/tutorials/arduino-ide-v1-basics)
* [MQTT and Lab4](https://github.com/ECE445L/ECE445L-Resources/blob/main/MQTT%20Guide.pdf)
* Valvano Section 3.3 on software style
* Valvano Section 3.4 on developing modular software
* Valvano Section 5.7 and 6.2 on periodic interrupts

# Starter Files

* Starter projects:
  + Lab 3 template provided on the GitHub Classroom repo.
  + Lab 4 template provided on the GitHub Classroom repo.

# Required Hardware

|  |  |  |  |
| --- | --- | --- | --- |
| Parts | Datasheet | Price | Source (**price source)** |
| EK-TM4C123GXL | [EK-TM4C123GXL datasheet](https://github.com/ECE445L/ECE445L-Lab5/blob/main/resources/TM4C_Datasheet.pdf) | $16.99 | **TI** |
| Sitronix ST7735R Color LCD | [ST7735R Driver datasheet](https://cdn-shop.adafruit.com/datasheets/ST7735R_V0.2.pdf) | $19.95 | **Adafruit** |
| 8Ω or 32Ω speaker | N/A | N/A | EER Checkout Desk |
| 10kΩ resistor | N/A | N/A | EER Checkout Desk |
| 1uF Tantalum Capacitor | N/A | N/A | EER Checkout Desk |
| Switches | N/A | N/A | EER Checkout Desk |
| IRLD120 MOSFET  or IRLD024 MOSFET | [IRLD120 datasheet](https://www.vishay.com/docs/91310/sihld120.pdf)  [IRLD024 datasheet](https://www.vishay.com/docs/91308/sihld24.pdf) | $1.88 | EER Checkout Desk  Or Mouser, **Digikey** |
| 1N914 Diode | [1N914 datasheet](https://www.vishay.com/docs/85622/1n914.pdf) | $0.10 | EER Checkout Desk,  Or **Mouser** |
| ESP8266-ESP01 | [ESP8266 Datasheet](https://www.espressif.com/sites/default/files/documentation/0a-esp8266ex_datasheet_en.pdf) | 3.50 | Get it from the TA’s |
| ESP01 breadboard adapter | N/A | N/A | Get it from the TA’s |
| ESP8266 Programmer |  | 10.99 | EER Checkout Desk |
| LM2937-3.3 Voltage Regulator **or** any 3.3V regulator | [LM3927 Datasheet](https://www.ti.com/lit/ds/symlink/lm2937.pdf?ts=1695011187491) | 1.68 | EER Checkout Desk |

# Lab Overview

A screenshot of a computer

AI-generated content may be incorrect.

Figure 4.1: Lab 4 Data Flow Diagram

Your team will upgrade the Lab 3 alarm clock with Wi-Fi to create a "smart object" with a web interface connected to it via an MQTT Broker. The TM4C lacks Wi-Fi, so we add the ESP8266 01S which will provide connectivity. The architecture is shown in Figure 4.1, with the right-side representing Lab 3 work and the left side showing the web interface and ESP8266.

In specific, you must be able to:

* Run the MQTT Web interface in a browser, displaying updates from the smart object. The web shows the current time (hour, minute, second) of the smart object.
* Have the smart object and its display respond to changes from the MQTT web interface. The web interface can send any of the following commands: Hour++, Hour--, Minute++, Minute--, Second++, Second--, and Toggle 12/24-hour display.
* Optionally, measure and display a calibrated measurement value on the web interface (extra credit). Any sensor and interface method (ADC, I2C, or SPI) is acceptable.

# Preparation

1. Edit the requirements document in the **Lab04Report.docx** to reflect your design. The requirements document is fluid - we expect it to change as you develop your solution. You should modify the requirements document.
2. Demonstrate that you can flash the ESP8266
   1. Download and install the [Arduino IDE](https://docs.arduino.cc/software/ide/). Then launch the program and navigate to File --> Preferences. In the Additional board manager URL’s, add the following URL:

<http://arduino.esp8266.com/stable/package_esp8266com_index.json>

A screenshot of a computer

AI-generated content may be incorrect.

Figure 4.2 Adding Board Manager URL

* 1. On the left bar, open the Board Manager and search for “ESP8266”. Install “esp8266 by ESP8266 Community”. After installing, connect the programmer to your computer via USB and select the board as a Generic ESP8266 module.

A screenshot of a computer

AI-generated content may be incorrect.

Figure 4.3 Installing the board manager

* 1. On the left bar, open the Library Manager. Search for and install both the “PubSubClient by Nick O’Leary” and “Blynk by Volodymyr Shymanskyy” as shown below.

A screenshot of a computer

AI-generated content may be incorrect.

Figure 4.4 Installing software dependencies

* 1. Now to program the ESP. Open the .INO file provided with the Arduino IDE. Plug the ESP into the programmer and the programmer into the laptop. Verify the proper board is selected in the IDE by going Tools -> Board -> ESP8266 -> Generic 8266 Module. Verify that the programmer is detected by navigating to Tools -> Port and selecting a valid option (You may need to [install CP210X drivers](https://www.silabs.com/developer-tools/usb-to-uart-bridge-vcp-drivers?tab=downloads) and restart the Arduino IDE). Within the Arduino IDE on the top left corner, press upload. If this fails to flash automatically, press the reset button on the programmer, then press D15 button twice to enter the ESP Bootloader and try again.

A screenshot of a computer

AI-generated content may be incorrect.

Figure 4.5 Configuring for flashing

1. Connect the ESP to a MQTT Broker via Wi-Fi
   1. Setup Wi-Fi sharing on a device, preferably connected to UTexas. In Windows go to Settings -> Network -> Mobile Hotspot -> Edit. Ensure 2.4Ghz is selected. Turn on the hotspot and disable power saving to prevent the hotspot from turning off automatically.

A screenshot of a computer

AI-generated content may be incorrect.

Figure 4.6 Wi-Fi Sharing

* 1. Now we need to select an MQTT broker. You may use any MQTT broker, but we recommend using McDermott’s if you are on campus. Note that if you connect the ESP to non-utexas Wi-Fi **YOU WILL NOT BE ABLE TO USE McDermott’s MQTT Broker**.
     1. McDermott: IP/Domain: 10.159.177.133, Port 9001 (WebSocket/TCP)
     2. EMQX: IP/Domain: broker.emqx.io, Port 8083 (WebSocket) 1883 (TCP)
     3. Selfhost: Setup your own broker, one option is [MOSQUITTO](https://mosquitto.org/)

1. Demonstrating bidirectional UART and Web Interface communication

# Procedure

## 

## Deliverable 1

Draw the electrical circuit you used to create the alarm clock. Put either a diode or a capacitor in parallel with the speaker. If you use a polarized tantalum capacitor, orient the + and - pins with the direction of the current flow. The diode on the other hand must be oriented with the + and - pins opposite with the direction of the current flow.

## Deliverable 2

Measure how long it takes the LCD graphics to update.

## Deliverable 3

Show the RMS noise level on the 3.3V with and without the alarm sounding. Using a voltmeter in AC mode, you should get a value of between 0.5-5mV.

## Deliverable 4

Measure the voltage versus time of the drain pin of the MOSFET, without capacitor, and use it to determine the current through the speaker. Measure the frequency of the sound. Place a picture of the scope trace like **Figure 3.3b** into your lab report, either a photo or digital downloaded image

## Deliverable 5

Measure the voltage versus time of the drain pin of the MOSFET, with the capacitor or the diode. Place a picture of the scope trace like **Figure 3.3c** or **Figure 3.3d** into your lab report, either a photo or digital downloaded image

## Deliverable 6

Show the system current with and without the alarm sounding. This is measuring the current through the entire system. You can do this by connecting the lab DC power supply (set to 5V and 500mA) to the TM4C’s VBUS and GND.

# Lab Checkout

The lab checkout is performed during the M/T lab session.

Demonstrate all the functions of your alarm clock including the required ones as well as any extra features. Additionally, show that your digital alarm clock is stand-alone by turning the power off, then on. The digital alarm clock should run (the time will naturally have to be reprogrammed) without downloading the software each time.

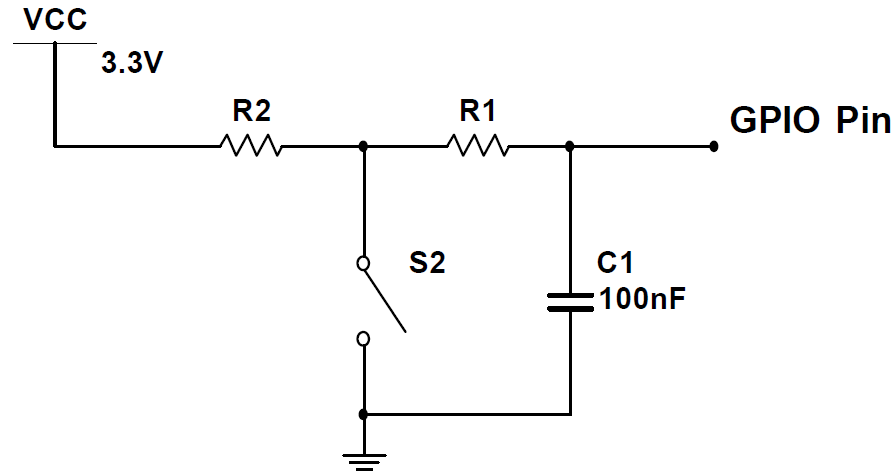
# Lab Report

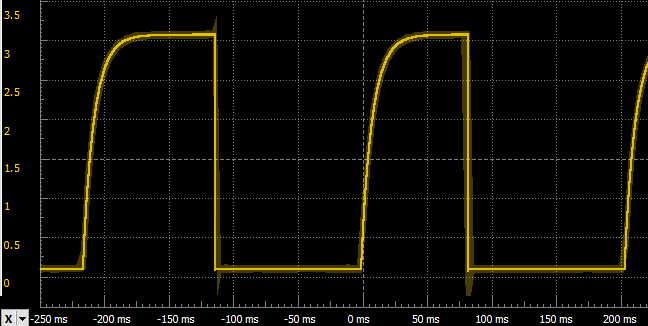
The lab report shall be submitted by the Friday after the second lab section.

You should complete the Lab03Report.docx file with your data and answers then submit the completed file to Canvas.

# Hints

1. The requirements document should change a couple of times during the lab as you determine features.
2. You can interface an 8Ω or 32Ω - speaker to an output port using an NPN MOSFET like the IRLD120. A 10kΩ resistor between digital output pin and gate reduces current surges but does not affect loudness. Loudness is determined by the voltage drop across the speaker. Connect the source to ground, and the drain to one side of the speaker. Connect the other side to +3.3V. The maximum of the transistor must be larger than 3.3V/8Ω or (3.3V/32Ω). The speaker has inductance, but the MOSFET includes an internal diode to remove back EMF when the transistor switches off. If you toggle the output pin in the background ISR, then sound will be generated. Loudness is determined by the voltage drop across the speaker. From Figure 3.3a, we see the MOSFET drain voltage is about 0.5V when active. So, the voltage drop will be 3.3V-0.5V = 2.7V
3. You must be careful not to let the LCD show an intermediate time of 1:00 as the time rolls over from 1:59 to 2:00. You must also be careful not to disable interrupts too long (more than one interrupt period), because a time error will result if any interrupts are skipped.
4. You may use 32-bit or 64-bit timer modes on the TM4C microcontrollers. However, it is good practice to refer to the errata for the microcontroller you are using. The errata describe bugs and flaws not listed in the data sheet.
5. If you use the on-board switches, then you must activate the internal pull-up resistors. You will set the corresponding bits in the GPIO\_PORTF\_PUR\_R register. These on-board switches are simply SPST switches to ground. When the switch is pressed, the signal goes to 0V (ground). When the switch is not pressed, the internal pull-up makes the signal go high (3.3V.) Furthermore, coming up out of a reset PF0 is locked, and thus if you use PF0 you will need to unlock it.
6. Learn to use KiCad. You will need to be proficient with this application during Labs 6 and 7. Using it now for simple circuits will be an efficient use of your time.
7. If you use edge-triggered interrupts, build an analog filter to debounce each switch. Set R1=0, and R2=100k to create the negative logic switch. Choose R2 and C1 so the time constant (τ=R2\*C1) is around 10 ms. Test the circuit with a scope before connecting to the microcontroller.





1. A better method to debounce edge-triggered interrupts is to use a second timer (see **EdgeInterruptDebounce\_4C123**)