Microcontrollers Lab - 10636496

Arduino Experiment #2: Serial Connection

Prepared by: DR. MANAR QAMHIEH

Objectives:

4-digit 7-segment LED display

A 4-digit 7-segment display is useful for any project involving a clock, timer, or counter without the need to use a display. TM1637 module uses a two-wire interface where it requires only four connections in total – two for power and two for controlling the display.

The TM1637 module combines a classic 0.36″ 4-digit 7-segment display and the TM1637 LED driver allowing you to control all digits, and the colon using only two I/O pins. The TM1637 handles all the work of refreshing the display after it has been updated by the microcontroller. This frees up the microcontroller to do other important things. Another plus is that the TM1637 has many useful features, such as the ability to adjust the display's brightness and control each segment independently.

The TM1637 module operates on a 3.3 to 5 volt supply voltage and communicates via a two-wire bus, requiring only two data pins, VCC and ground. The TM1637 has its own proprietary data transfer protocol, but there are Arduino libraries available that hide the complexities and make communication with the display easier.

- CLK is the clock input pin
- DIO is the Data I/O pin

The module does not rely on any pin-specific features, it is safe to use any two pins from the Arduino. Just make sure to update the pin numbers in the code to reflect any wiring changes.

To communicate with the TM1637 chip, you'll need to use a library. The **TM1637Display** library by Avishay Orpaz is an excellent library. This library includes a number of built-in functions to easily control the display.

```
#include <TM1637Display.h>

#define CLK 3         // based on your connection

#define DIO 4

TM1637Display display = TM1637Display(CLK, DIO);
```

Then open TM1637Display.h file for more information about the functions available in this library and especially:

`setBrightness, showNumberDec, showNumberDecEx`

These function can be displayed by using the display object.

For example: `display.setBrightness(7);`

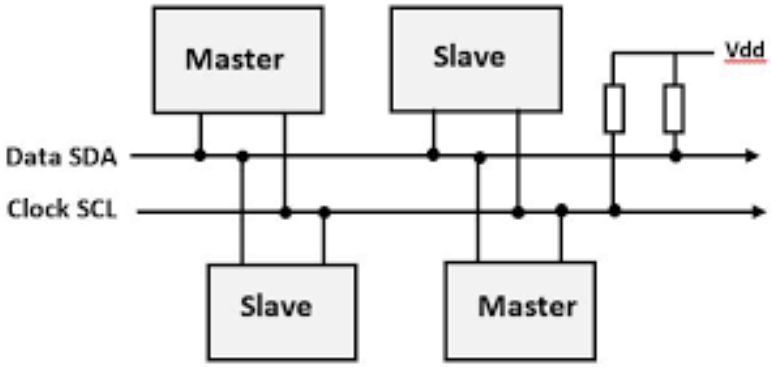
Task1: Write a code to display a 0-9999 BCD counter on the 4-digit LED display

Task2: Write a code to display a clock timer on the display to count minutes and seconds. Each count from 0 to 59. The seconds are displayed on rightmost digits and the minutes on the leftmost digits.

I²C Serial Communication

The I²C stands for Inter-Integrated Circuit. It is a bus interface connection protocol incorporated into devices for serial communication. It uses only 2 bi-directional open-drain lines for data communication called SDA and SCL. Both these lines are pulled high.

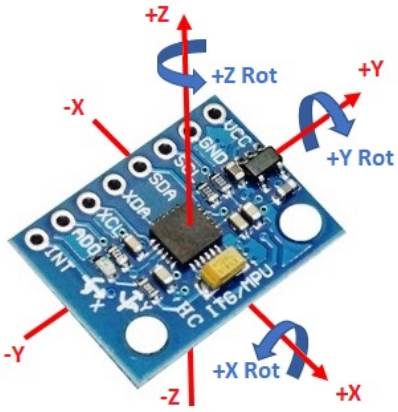
- Serial Data (SDA) – Transfer of data takes place through this pin.
- Serial Clock (SCL) – It carries the clock signal.



If you have multiple devices on the same I²C bus, you may need to set a different I²C address to avoid devices conflicting with each other.

Accelerometer- gyroscope GY-521 (datasheet, schematic)

The MPU6050 contains both a 3-Axis Gyroscope and a 3-Axis accelerometer allowing measurements of both independently, but all based around the same axes, thus eliminating the problems of cross-axis errors when using separate devices.



This simple module contains everything required to interface to the Arduino and other controllers via I²C (use the Wire Arduino library) and give motion sensing information for 3 axes - X, Y and Z.

The GY-521 module is a breakout board for the MPU-6050 MEMS (Microelectromechanical systems) that features a 3-axis gyroscope, a 3-axis accelerometer, a digital motion processor (DMP), and a temperature sensor.

The MPU-6050 sensor values are retrieved by using the I^2C serial data bus, which requires only two wires (SCL and SDA).

- VCC (The breakout board has a voltage regulator. Therefore, you can connect the board to 3.3V and 5V sources.)
- GND
- SCL (Serial Clock Line of the I^2C protocol.) - pin 21 on Arduino Mega
- SDA (Serial Data Line of the I2C protocol.) - pin 20 on Arduino Mega
- XDA (Auxiliary data => I2C master serial data for connecting the module to external sensors.)
- XCL (Auxiliary clock => I2C master serial clock for connecting the module to external sensors.)
- AD0 (If this pin is LOW, the I2C address of the board will be 0x68. Otherwise, if the pin is HIGH, the address will be 0x69.)
- INT (Interrupt digital output)

We will connect only the power, SCL and SDA pins.

Task 3: Write a code to detect the tilting of the module over the x and y axes.

Hints:

- make use of the Arduino platform's in-built library (Wire) to establish an I^2C connection between the Arduino Uno and the GY-521 sensor.

```
#include <Wire.h>
```

- Initialize the module by:

```
Wire.begin();
Wire.beginTransmission(MPU_ADDR);     // MPU address 0x68
Wire.write(0x6B);                     // PWR_MGMT_1 register
Wire.write(0);                        // set to zero (wakes up
```

```
// the MPU-6050)
Wire.endTransmission(true);
```

- In the loop function, seven sensor values (3x accelerometer, 1x temperature, and 3x gyro) are requested from the GY-521 module.
- Whenever you want to read data from the module, you send:

```
Wire.beginTransmission(MPU_ADDR);
Wire.write(0x3B); // starting with register 0x3B (ACCEL_XOUT_H)
Wire.endTransmission(false); // the parameter indicates that the
// Arduino will send a restart. As a result, the connection is
// kept active.
Wire.requestFrom(MPU_ADDR, 14, true);      // request a total of
// 7*2=14 byte-registers
```

- To read the 16-bit values, you need to do the following:

```
Int raw_ax = Wire.read()<<8 | Wire.read();

// reading registers: 0x3B (ACCEL_XOUT_H)
// and 0x3C (ACCEL_XOUT_L) (means two registers are read and
// stored in the same variable)
```

- To write a tilting application, read 3 values raw_ax , raw_ay and raw_az. If you need the temperature, it is the fourth value.
- For a better view of data, you need to convert it by:

$$requiredValue = \frac{rawValue}{sensitivity}$$

Where the sensitivity is either 2048, 4096, 8192 and 16384

- You need to analyze the read values and write proper application to identify the tilting of the module (either towards the top/bottom - y-axis or left/right - x-axis).
- To print the temperature, you need to convert it first as follows:

$$Degree = \frac{temperature}{340} + 36.53$$

Liquid Crystal Display with I²C

The LCD is an informative output that can easily show characters on its screen. LCDs range in size, price and configuration, from having a couple of lines to large displays. Some are even very specifically designed for a single application, having only that ability to display set graphics. In this lab, we will use a 16x2 LCD which consists of two lines each has 16 characters.

The most basic pins of standard LCD are the power pins for the display to be able to function in the first place. There is a **VDD** pin for 5 volts and a **VSS** pin for ground. There is a **V0** pin for the adjustment of the LCD contrast. Some LCDs even have an LED backlight and are generally the last two pins. The LCD has a row of 8 pins to serve as its port which are **D0, D1, D2, D3, D4, D5, D6 and D7**. These pins are generally used to pass information into the LCD, but it can also be set to pass information back to the microcontroller.

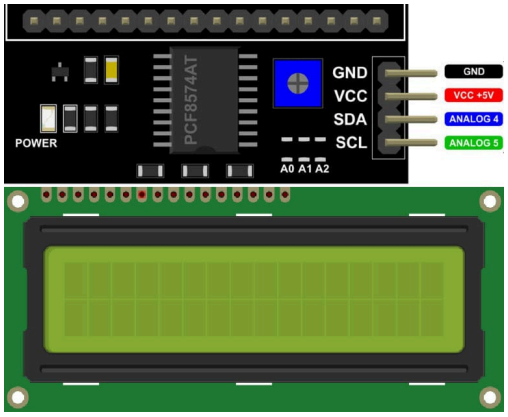
Two types of information can be passed through these pins:

- data to display on the LCD screen
- control information such as clearing the screen, controlling the cursor position, turning the display on or off, etc.

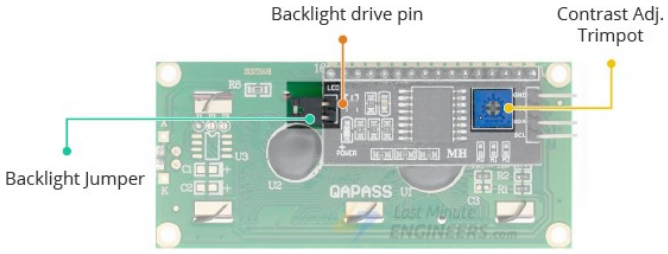
The pin on the LCD that is responsible for the read and write state is labeled **R/W**. The pin on the LCD that is responsible for whether the infomation sent is a character or a control, is the **RS** pin (Register Select). And the pin that helps the LCD accept informatin is called the **EN** pin (Enable).

Standard LCD vs. I²C

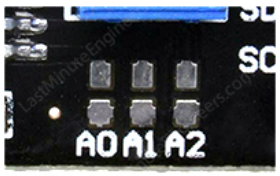
The standard LCD has a parallel interface, meaning that the microcontroller has to manipulate several interface pins at once to control the display as explained above. The **I²C LCD** is actually a standard parallel LCD that has a separate I2C adapter board that allows the LCD to use a serial data bus by using 4 pins only (two of them for power).



At the heart of the adapter is an 8-bit I/O expander chip – PCF8574. This chip converts the I2C data from an Arduino into the parallel data required for an LCD display.The board also includes a tiny potintiometer for making precise adjustments to the display's contrast and a jumper on the board that provides power to the backlight.



To set the I^2C address, the adapter comes with three solder jumpers/pads (A0, A1, and A2). The address is set when a jumper is shorted with a blob of solder. The I^2C address of your LCD depends on the chip manufacturer.



If all three address inputs are pulled HIGH using onboard pullups (as done in the adapters we use in the lab), this gives the PCF8574 a default I2C address of **0x27**.

The I^2C LCD Display has only four pins. The following is the pinout:

- SDA is the I^2C data pin connected to the SDA pin on Arduino board.
- SCL is the I^2C clock pin connected to the SCL pin on Arduino board.

Task 3: Display Hello message on the first line and your name on the second line.

In order to use the I2C LCD, you need to install a special library called LiquidCrystal_I2C by Frank de Brabander and check its examples to see how it is used and the provided functions such as initialization, printing, setting the cursor and backlight.

Task 4: Display distance on LCD by using the ultrasonic sensor.