EECS 388 Lab #4

Sensor Reading: TFmini LiDAR

In this lab, we will attach a small LiDAR (light detection and ranging) sensor, which measures distance, to the Arduino board via UART and develop a program to read the sensor data.

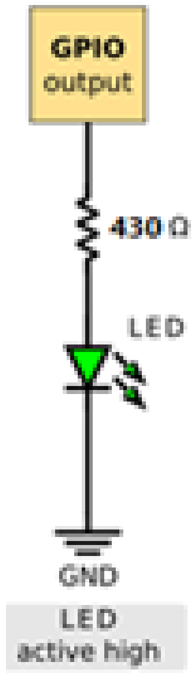
Task 0: Download starter code, setup circuit, read LiDAR documentation

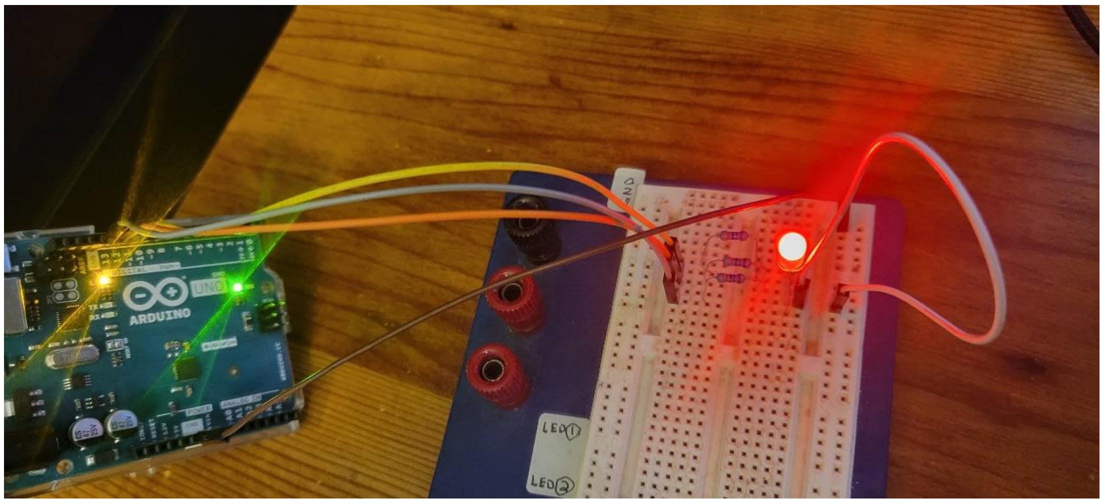
Task 0.0 Download starter code

Log into Canvas and go to the EECS 388 course page. Go to “Files>Labs” and under “Lab 4” click on **lab04.zip** to download the lab source code. Extract the files into a folder on your PC and open the folder in **VSCode**, same as with Lab 2 and 3.

Task 0.1 Setup LED circuit from lab 2 and 3

Setup the same active-high GPIO LED circuit that you used with lab 2 and 3. Refer to the picture and schematic for setup. Once you have the circuit setup, review it with your GTA before turning on the Arduino Uno.





Task 0.2 Read LiDAR documentation and review Arduino serial documentation

The tfmini LiDAR datasheet can be found in **lab04/docs/ benewake-tfmini-datasheet.pdf**. Be sure to look at **Section 1 – Product Characteristics**. Note that the operating range of 0.3m to 12m. The LiDAR's minimum operating range is 0.3m or ~12 inches. Below this, it will have difficulty detecting the distance. Also review **Section 6 – Serial Port Data Communication Protocol.** Note the required baud rate, data bits, stop bits, and parity bits that we need to setup on the Arduino to connect with the tfmini LiDAR. **What baud rate does the tfmini LiDAR use**?

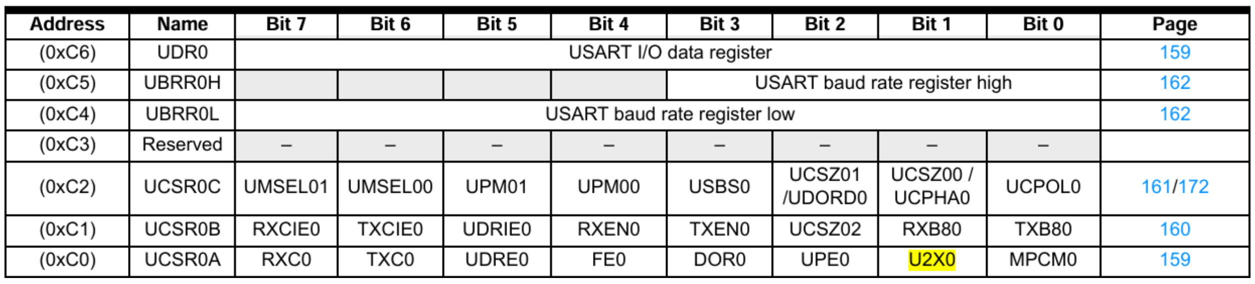
The data exchange format of the sensor is described in **Section 6.1** "Standard Data Format of Serial Port" of the datasheet. The simplified version of Table 6 of Section 6.1 is shown below.

Byte 1	2	3	4	5	6	7	8	9
0x59	0x59	Dist_L	Dist_H	Strength_L	Strength_H	RSVD	Quality	Chksum

In this sensor, each measured data is encoded in a 9-byte data frame. The first two bytes are frame headers (0x59 = 'Y' in ASCII). The next two bytes encode the actual distance data (**<u>in centimeters</u>**). Note that byte 3 (Dist_L) is the low 8 bits and byte 4 (Dist_H) is the high 8 bits of the measured distance data (16 bit). The rest of the bytes of a data frame are not used in this lab.

Note that the sensor operates at 100 Hz, generating one data frame (distance measurement) every 10 ms.

Also review chapter 19 in the Arduino datasheet **lab04/docs/ Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf** for information about the UART. In this lab we will need to modify uart_init() so take a look at the data and control register values, specifically the U2X0 bit in Control Register C.



Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
(0xC6)	UDR0	USART I/O data register								159
(0xC5)	UBRR0H					USART baud rate register high				162
(0xC4)	UBRR0L	USART baud rate register low								162
(0xC3)	Reserved	–	–	–	–	–	–	–	–	
(0xC2)	UCSR0C	UMSEL01	UMSEL00	UPM01	UPM00	USBS0	UCSZ01 /UDORD0	UCSZ00 / UCPHA0	UCPOL0	161/172
(0xC1)	UCSR0B	RXCIE0	TXCIE0	UDRIE0	RXEN0	TXEN0	UCSZ02	RXB80	TXB80	160
(0xC0)	UCSR0A	RXC0	TXC0	UDRE0	FE0	DOR0	UPE0	U2X0	MPCM0	159

- **Bit 1 – U2Xn: Double the USART Transmission Speed**

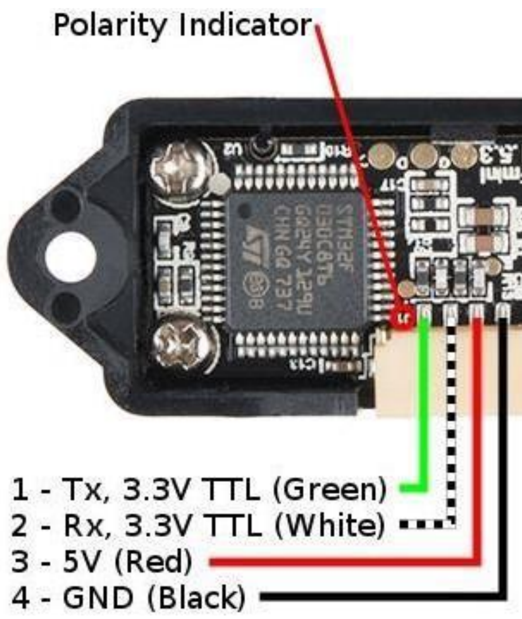
This bit only has effect for the asynchronous operation. Write this bit to zero when using synchronous operation.

Writing this bit to one will reduce the divisor of the baud rate divider from 16 to 8 effectively doubling the transfer rate for asynchronous communication.

Task 1: TFmini LiDAR

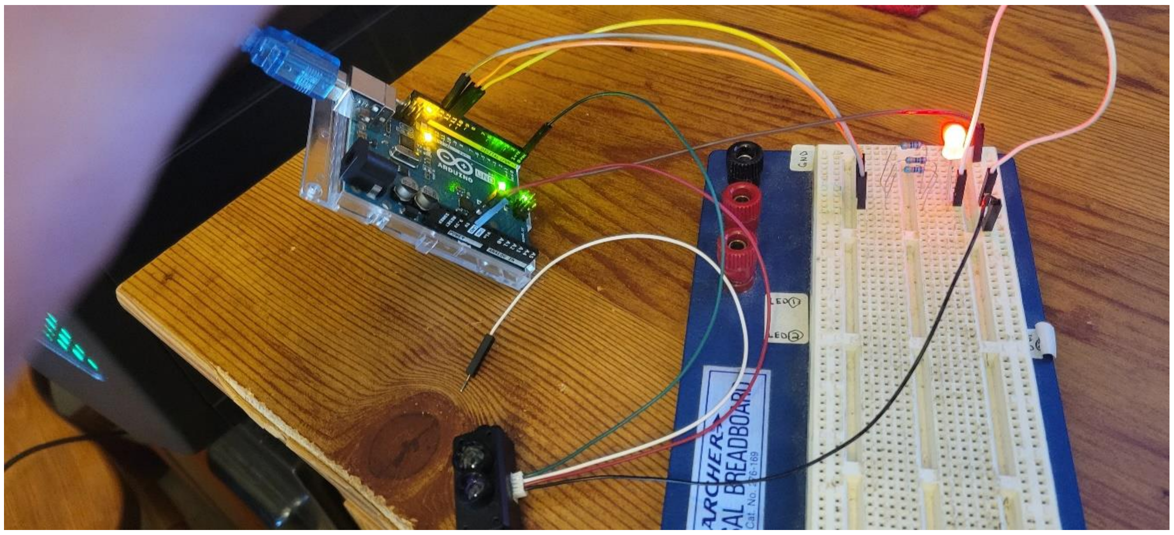
Task 1.0 Connect the LiDAR sensor to your board

The TFmini sensor can be connected to the Arduino board via a UART connection.



You need to connect the TFmini's Tx (green) to the Arduino board's UART RX (pin 0). You do not need to connect the TFmini RX line, as we don't need to transmit anything to the LiDAR sensor. The 5V (red) and GND (black) should be connected to Arduino's 5V and GND lines, respectively. Review your connections with your GTA before powering on the board.

The picture below shows the final connections.



Task 1.1. Update uart_init() baud rate and double speed mode

Use the formula from Lab 3 to calculate the UBRR value for the baud rate required by the LiDAR sensor.

$$\mathrm{UBRR} = \frac{\mathrm{f_{osc}}}{16 \times BAUD} - 1$$

Before we use that value in task 1.1, let's calculate the %error. We use this formula to calculate the error. To get the actual_baud, use the calculated UBRR value, f_{osc} = 16,000,000 (16MHz), and solve for BAUD.

% Error = (absolute_value(actual_baud – desired_baud) / desired_baud) * 100%

What % Error did you calculate? For effective UART communication, we typically want an acceptable error rate based on our application:

- <= 2% //Typically reliable
- 3%-5% //May be okay depending on noise, signal quality, and baud rate (example: may work at a lower baud rate, but not a higher baud rate)
- >5% //Not reliable, may have framing errors, corrupted bytes, or missed packets especially for higher baud rates

Since our error rate is greater than 3% our communication with the LiDAR sensor might not be reliable! Luckily the ATMega328P has a way to reduce the error rate: double speed mode.

To put the ATMega328P into double speed mode, we need to set the U2X0 bit in the UCSR0A register. Update your uart_init() code to set the flag.

If we put the UART into double speed mode, the divisor for UBRR becomes 8 instead of 16. Repeat the process above to calculate the new UBRR value.

UBRR = (f_{osc} / (8 x BAUD)) - 1

Verify the new % Error. Update your uart_init() code by initializing ubrr to the calculated value.

Task 1.2. Implement the code to read the TFmini data frame

You will re-use the `ser_read()` function you implemented in Lab 3 here (or use the provided `ser_read()` function in the source code you downloaded).

The first task is to interact with the TFmini sensor via UART0 to parse the sensor's distance data. Note that the distance data is stored in bytes 3 and 4 of the sensor's data frame. So, you need to do a bit of data manipulation to obtain the distance value. Note that the variable **dist** should contain the obtained distance value.

If needed, you print the value to the console (i.e. the serial monitor), by using either `ser_printf()`, `ser_printline(), and ser_write()` functions. **Note** however on the Arduino that the RX and TX buffers are shared, so if you are writing, it may interfere with receiving from the TFMini LiDAR.

Task 1.3. Control LED based on distance

The next task is to control the red and green LEDs based on the observed distance. If the distance between the sensor and the object is less than 75 cm, turn on the red LED. Otherwise, turn on the green LED.

Task 1.4. – Confirm the checksum – Optional Bonus – 1 additional point

Calculate the checksum by receiving all 9 bytes of data that are transmitted by the TFmini LiDAR.

calculated_checksum = Byte1 + Byte2 + Byte3 + .. + Byte8

Compare the least significant byte of the calculated_checksum with the received Byte9 and confirm they match.

Submission – 40 points (Max 42/40 with bonus)

Milestone: To obtain the 10% (4 points) for this lab, you must have the TFmini LiDAR correctly connected to the board, and show that you are receiving some data in the terminal window.

Demo: To obtain the 40% (16 points) for this lab, you must show your GTA the working LiDAR sensor, when detecting a distance less than 75cm, the LED should be turned red. For distances greater than 75cm, the LED should be turned green. You will also need to be able to answer some of the following questions:

Possible questions:

- What is the baud rate for this UART connection?
- What was the original error rate in single speed mode?
- What is the new error rate in double speed mode?
- How many stop bits are used? 1 or 2?
- Is it even, odd, or no parity bit?

- What is the maximum distance you can read if you only receive the Dist_L (lower byte) from the TFmini?
- If there were 3 bytes of distance data, how would you put the total distance together?
- What is the operating range of the TFmini LiDAR?

Code: To obtain the 50% (20 points) go to the "Assignments" section on Canvas and submit your modified **eecs_388_tfmini.cpp and eecs_388_lib.c** files to Lab 04 submission link. Also, make sure that you have shown your completed work to your respective GTA for a demo.

Screenshots/PDF/Text files will **NOT** be allowed/graded as submissions. You need to submit a modified **eecs388_tfmini.cpp and eecs_388_lib.c** files **only**.

Bonus:
To obtain the 2.5% (1 points) you need to submit the files to Canvas by the end of your lab period. You must demo your working code on the board to your GTA before submitting the code to Canvas.

You can also obtain an additional 2.5% (1 point) by implementing the bonus task of checking the calculated and received checksum values and demo it to your GTA.

Appendix – Debugging Tips and Tricks

- **There is only 1 UART on the Arduino UNO board, it is used for both the TFmini LiDAR and for programming. If you see errors uploading your code, you may need to remove the green wire from the Arduino Uno board, upload your code, and then plug it back in.**
- The TFmini LIDAR can get into an error state or provide unreliable distances. If you disconnect the USB cable from the PC and plug it back in, it will "reboot" the Arduino Uno and the TFMini LiDAR and may get it working again
- Try using ser_printf() like the printf() function to debug if you need to print to the console
- Remember the UART RX/TX is shared, so if you are printing/using the buffer it might having timing conflicts receiving from the TFminiLidar