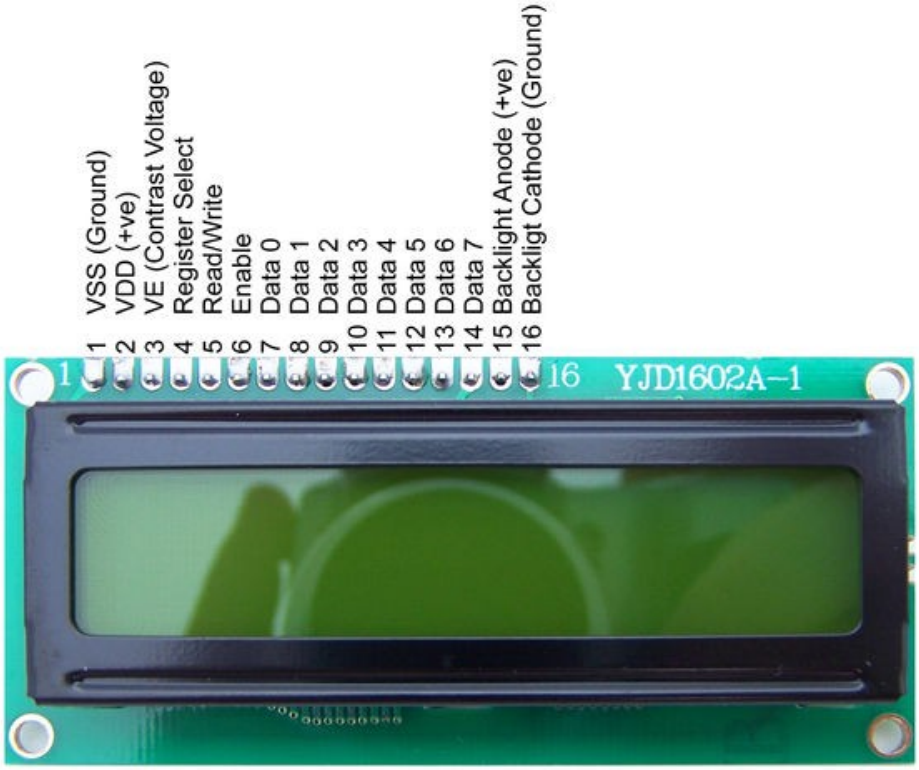
Lab Number 4

In this lab you will make use of the ultrasonic sensor circuit, the LCD display and the servo used in lab number 1. Building on the previous three labs, you will put all of them together to create a simple “radar” system. The system will also incorporate custom characters to display the location information of detected objects. Using a small breadboard, you will mount the ultrasonic sensor on top of the servo so that it can be positioned in any direction.

The system should keep track of at most five objects in the scanned area and display their distance and relative location.

The LCD connection information from lab number 3 is repeated below:

The image and wiring table below show the connections to be made and the superscript numbers in the table show the order of the pins to be used in the LCD function call.



GND	+5V	Contrast	Reg Sel[1]	Read / Write	Enable[2]	D0	D1	D2	D3	D4[3]	D5[4]	D6[5]	D7[6]	Bklight[7]+3.3V	GND
Main Power for display		Display Contrast. Should go to the center pin of a potentiometer	Command when high and data when low	Always GND or voltage must be controlled	Used to decide when to send data to the display	No Connection	No Connection	No Connection	No Connection	Data	Data	Data	Data	This can be 5V or 3.3V (less bright)	GND for backlight

Remember, to use the LCD, you must import the library used in lab number 3.

Calls from that library are listed below.

import myLCD

start(GPIO,rows, cols) will be used to define the size of the display

LCD(GPIO,RS[1], E[2], D4[3], D5[4], D6[5], D7[6], BKL[7]) is used to define the pins that will be used for each of the lines in the table above.

lcd_init(GPIO) is used to clear the screen, and put it into a state ready to accept input.

Now you can use any of the following functions to display your data and control the LCD as needed.

backlight(GPIO,state) is used to turn the back light on (True) or off (False)

lcd_string(GPIO,message, line) is used to display text on the screen

lcd_shiftleft(GPIO)

lcd_shiftright(GPIO) shift the display left and right by one character

lcd_shutdown(GPIO) is used to clear the display, and clean up any other LCD properties that were set

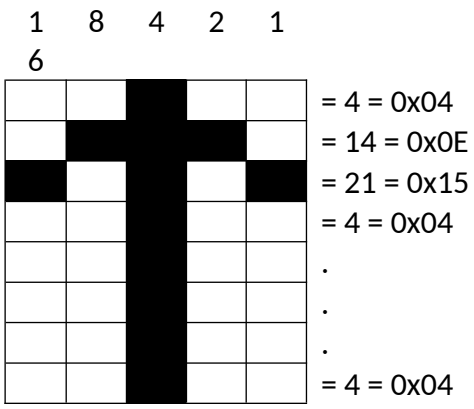
lcd_init(GPIO) can also be used the clear the screen at any time it becomes necessary.

In order to use custom characters, we will use a table with 5 columns and 8 rows. This is how each character is displayed on the LCD. You will create a list with the value of each row stored in hexadecimal.

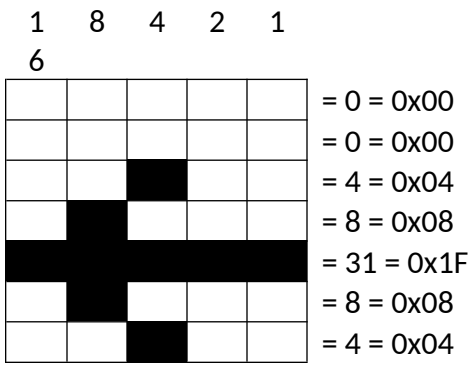
To get the numbers, use the following:

0 – 9, 10 = A, 11 = B, 12 = C, 13 = D, 14 = E, 15 = F

If the number is 16 or above, subtract 16 and use 0x1n where n is the amount left over. Or in this case, if the left column is filled, it will be 0x1n and you can add the rest without doing the subtraction. As examples, two characters have been defined below.



For the character above, upArrow=[0x04, 0x0E, 0x15, 0x04, 0x04, 0x04, 0x04, 0x04] would define it



| | | | | | = 0 = 0x00

For the character above, leftArrow=[0x00, 0x00, 0x04, 0x08, 0x1F, 0x08, 0x04, 0x00]

You can define up to 8 custom characters to be displayed on the LCD. The must be stored in the character table in character positions 0 to 7. These have been set aside for this reason.

Once defined, you will add the following function to the myLCD library. It can then be called to write the characters to your LCD memory.

```
LCD_CHARS = [0x40, 0x48, 0x50, 0x58, 0x60, 0x68, 0x70, 0x78]

def lcd_custom(GPIO, charPos, charDef):
    lcd_byte(GPIO, LCD_CHARS[charPos], LCD_CMD)
    for line in charDef:
        lcd_byte(GPIO, line,LCD_CHR)
```

In your program, you can now use something similar to:
lcd_custom(GPIO,0,upArrow)
lcd_custom(GPIO,2, leftArrow)

Now you can use the chr() function to display the characters you have created.
Ex. lcd_string(GPIO,char(0)+" "+char(1)+" "+char(2)+ " "+ …+" " + char(7),1)

You will use your custom characters to display the relative location of the objects detected as your "radar" sweeps across its possible range. Sample output can be seen below. You can add the distance for the objects to the second line if you wish.