Raspberry Pi

Lab 4

RADAR

You have three lab sessions to finish!

In this lab you will make use of the ultrasonic sensor circuit, the LCD display and the servo used in Lab 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.

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

LCD pin	**Description**	**Connection**
Vss	Main Power for display	Gnd or -5V
Vdd	Main Power for display	+5V
V_0	Display Contrast	Should go to the center pin of a potentiometer*
RS	Command when high and data when low	Raspberry Pi pin
RW	If it is high we read from LCD and if it is low we write to LCD	Gnd or -5V
E	Used to decide when to send data to the display	Raspberry Pi pin
D_0	Data bit 0	No Connection
D_1	Data bit 1	No Connection
D_2	Data bit 2	No Connection
D_3	Data bit 3	No Connection
D_4	Data bit 4	Raspberry Pi pin
D_5	Data bit 5	Raspberry Pi pin
D_6	Data bit 6	Raspberry Pi pin
D_7	Data bit 7	Raspberry Pi pin
A	Backlight positive pin	Raspberry Pi pin or +5V
K	Backlight negative pin	Gnd or -5V

Remember, to use the LCD, you must import the library used in lab number 3. To use the myLCD library, you should save it in the same directory with your code and then import it with the following command: **from myLCD import *** . Before you use any function from myLCD library you should have GPIO imported and set for BCM mode. You would then do the following to initialize the LCD:

- start(GPIO,rows, cols) will be used to define the size of the display
- LCD(GPIO,RS, E, D_4,D_5, D_6, D_7, BKL) 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 an initialized state.

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, the message variable is type of string and line can be 1 to use the first row or 2 to use the second row of the LCD.
- 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 be used the clear the display at any time.

2) Connect the motor to the breadboard.

Caution: Reversing the current through a servo motor can cause damage to the motor. Please take care when making these connections.

1) The red wire should be connected to the +5v column of the breadboard
2) The black wire should be connected to the -5v column of the breadboard
3) The white wire (this wire may be any color other than red or black but is usually white or yellow) is the control wire and should be connected to any numbered pin of raspberry. **Do not connect to any pin labeled 5V or 3V3.**

3) Connect the sensor

The VCC pin of the sensor should be connected to the 5V(+) rail, GND should be connected to the 5V(-) rail and the Trig pin should be connected to one of the numbered pins on the Raspberry Pi. This pin should be configured as output when writing your program. The Echo pin through the voltage divider should be connected to another of the numbered pins and it should be configured as input within your program.

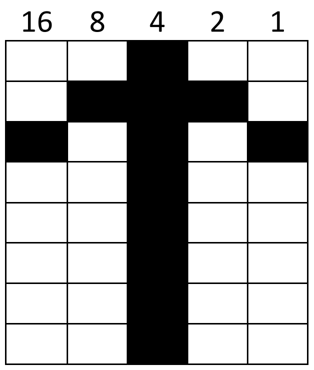
4) Create custom characters

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:

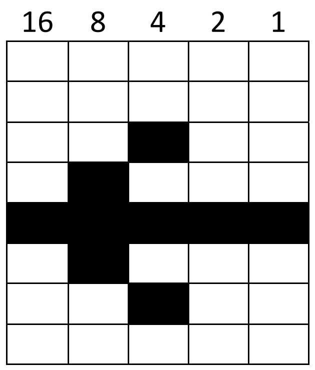
0 – 9, 10 = A, 11 = B, 12 = C, 13 = D, 14 = E, 15 = F

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.



= 4 = 0x04
= 14 = 0x0E
= 21 = 0x15
= 4 = 0x04
.
.
.
= 4 = 0x04

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



= 0 = 0x00
= 0 = 0x00
= 4 = 0x04
= 8 = 0x08
= 31 = 0x1F
= 8 = 0x08
= 4 = 0x04
= 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.

5) Write the code for the new characters

- **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:

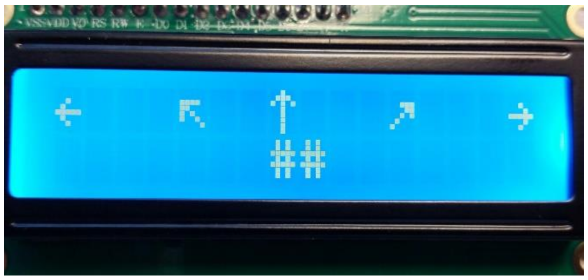
```
upArrow=[0x04, 0x0E, 0x15, 0x04, 0x04, 0x04, 0x04, 0x04]
leftArrow=[0x00, 0x00, 0x04, 0x08, 0x1F, 0x08, 0x04, 0x00]
lcd_custom(GPIO,0,upArrow)
lcd_custom(GPIO,2, leftArrow)
```

By these two commands you store the upArrow to the first position and the leftArrow to the third. Now you can use the chr() function to display the characters you have created by giving as argument the position of the character in the table. For example if you want to display the two arrows at the first row of the LCD separated by two spaces you have to write:

```
lcd_string(GPIO,chr(0)+"  "+chr(2),1)
```

6) Lab Assignment

The system should keep track of at most five objects in the scanned area and display their distance and relative location. For that you have to create the rest of the arrows shown in the next image.



You will use your custom characters to display the relative location of the objects detected as your “radar” sweeps across its possible range. You can add the distance for the objects to the second line if you wish.

HINT: You need two loops. The first loop rotates the servo from 0 degrees to 180 and the second from 180 to 0. Inside the loops you trigger the ultrasonic sensor (same code with Lab 3). Also you need to check the range and display the appropriate symbol at the right position if you detect an object. You have to update your LCD whenever you detect a new object and clear it when you reach far left or far right.