Appendix

Deepam Priyadarshi 19BPS1117 *

August 30, 2021

A LCD Interfacing

A.1 Updating the FLOSS-Arduino firmware

In order to interface LCD module with Arduino we need to introduce the LiquidCrystal library in the FLOSS-Arduino firmware through

`#include <LiquidCrystal.h>`

command at the beginning of the file.

Then we need to declare a variable `lcd` of `LiquidCrystal` type with the pin numbers of Arduino connected to LCD's rs, rw, enable, d4, d5, d6 and d7, as its parameters. In this experiment we have used 12,10,11,4,5,6 and 7 digital I/o pins to conect LCD's rs, rw, enable, d4, d5, d6 and d7 pins.

`LiquidCrystal lcd(12,10,11,4,5,6,7);`

Below is the `else if` block to receive the serial input codeword from the python console to carry out the interfacing task.

```
//case L -> LCD
  else if(val == 76){
    while(Serial.available()==0) {};
    val = Serial.read();
    if (val == 115){//'s'-> size of lcd, columns and rows
      int cols = Serial.read();
      int rows = Serial.read();
      lcd.begin(cols, rows);
    }
    if (val == 116){//'t'-> received text message
      lcd.clear();
      charCount = 0;
      while(true){
```

*Under the guidance of Dr. Subbulakshmi P, Vellore Institute of Technology, Chennai

```
14          while(Serial.available()==0){};
15          char c = Serial.read();
16          if(c ==10) {break;}
17          lcd.write(c);
18          charCount++;
19        }
20        if(charCount > 16){
21          delay(800);
22          for(int i=16; i<charCount; i++){
23            lcd.scrollDisplayLeft();
24            delay(500);
25          }
26          delay(800);
27          lcd.home();
28        }
29      }
30      val =-1;
31    }
```

A.2 Updating Python Arduino Toolkit

In order to interface the LCD we need to initialize the LCD with its type, which is its number of columns and row. Below is the function definition that I have added in the `class Arduino` of the toolkit.

```
1     def cmd_lcd_size(self, cols, row):
2         cmd=""
3         cmd = "L"+"s"+chr(cols) + chr(row)
4         self.ser.write(cmd.encode('utf-8'))
```

The above function sends the code word in ASCII format `76 115 16 2` for initializing a 16x2 LCD.

```
1     def cmd_lcd_text_display(self, text):
2         cmd=""
3         cmd="L"+"t"+text+"\n"
4         self.ser.write(cmd.encode('utf-8'))
```

The above function sends the input text in its ACII code value.

A.3 Python main() function

This code calls the functions declared in the Python Arduino toolkit in order to initialize the Arduino as well as start interfacing the LCD.

```
import os
import sys
cwd = os.getcwd()
(setpath, Examples) = os.path.split(cwd)
sys.path.append(setpath)

from Arduino.Arduino import Arduino
from time import sleep

class LCD_DISPLAY:
    def __init__(self, baudrate):
        self.baudrate = baudrate
        self.setup()
        self.run()
        self.exit()

    def setup(self):
        self.obj_arduino = Arduino()
        self.port = self.obj_arduino.locateport()
        self.obj_arduino.open_serial(1, self.port, self.baudrate)

    def run(self):
        self.lcd_col, self.lcd_row = [int(x) for x in
        ↪  input("Enter the no. of columns and rows of LCD:
        ↪  ").split()]
        self.obj_arduino.cmd_lcd_size(self.lcd_col, self.lcd_row)
        self.text = ""
        while(True):
            self.text = input("Enter the text message: ")
            self.obj_arduino.cmd_lcd_text_display(self.text)
            sleep(1)
            c = input("Do you want to continue?(y/n)")
            if(c != "y"):
                break
```

```
33
34      def exit(self):
35          self.obj_arduino.close_serial()
36
37  def main():
38      obj_lcd = LCD_DISPLAY(115200);
39
40  if __name__=='__main__':
41      main()
```

B HC-SR04 Interfacing

B.1 Updating the FLOSS-Arduino firmware

The current firmware doesn't have code to measure the time duration for pulse being high on a pin. The below mentioned code introduces the `pulseIn()` function, an Advanced I/o function Arduino which take pulse input from a I/o pin.

If the Arduino read a ASCII value 80 (corresponding to char 'P') on its serial port, then it executes the below mentioned `else if` block.

```
1   //case P -> reading pulse input
2   else if (val==80){
3     while(Serial.available() == 0){};
4     val = Serial.read();
5     if (val > 48 && val < 102){
6       pin = val-48; // pin for reading the pulse
7       while(Serial.available() == 0){};
8       val = Serial.read();
9       if(val == 48){// 0 -> reading low pulse input
10        float duration = pulseIn(pin, LOW);
11        Serial.println(duration);
12      }
13      if(val == 49){// 1 -> reading high pulse Input
14        float duration = pulseIn(pin, HIGH);
15        Serial.println(duration);
16      }
17    }
```

```
    val = -1;
  }
```

B.2 Updating Python Arduino Toolkit

The toolkit need to be updated in order to activate the above mentioned `pulseIn()` function.

Below is the `python function cmd_pulse_in` which has been added to the `class Arduino` of the toolkit. It takes the pin number and the logical value for that pin as its input.

```
def cmd_pulse_in(self,ard_no,pin, val):
    a = ["0", "1", "2", "3", "4", "5", "6", "7", "8", "9", ":",
    ↪  ";", "<", "=", ">", "?", "@", "A", "B", "C", "D"]
    cmd = ""
    cmd = "P" + a[pin] + str(val)
    self.ser.write(cmd.encode('utf-8'))
    a=self.ser.read_until()
    a=a.decode('utf-8')
    return(a.split("\r")[0])
```

B.3 Python main() function

This code calls the functions declared in the Python Arduino toolkit in order to initialize the Arduino as well as start interfacing the HC-SR04 along with LCD.

```
import os
import sys
cwd = os.getcwd()
(setpath, Examples) = os.path.split(cwd)
sys.path.append(setpath)

from Arduino.Arduino import Arduino
from time import sleep

class LED_ON:
    def __init__(self, baudrate):
        self.baudrate = baudrate
```

```
13          self.setup()
14          self.run()
15          self.exit()
16
17      def setup(self):
18          self.obj_arduino = Arduino()
19          self.port = self.obj_arduino.locateport()
20          self.obj_arduino.open_serial(1, self.port, self.baudrate)
21
22      def run(self):
23          self.echoPin = 2
24          self.trigPin = 3
25          self.obj_arduino.cmd_lcd_size(16,2)
26          while True:
27              self.obj_arduino.cmd_digital_out(1, self.trigPin, 0)
28              sleep(0.000001)
29              self.obj_arduino.cmd_digital_out(1, self.trigPin, 1)
30              sleep(0.000002)
31              self.obj_arduino.cmd_digital_out(1, self.trigPin, 0)
32              dist =
                ↪  round(float(self.obj_arduino.cmd_pulse_in(1,self.echoPin,
                ↪  1)) * 0.034 / 2)
33              sleep(0.7)
34              self.obj_arduino.cmd_lcd_text_display("Dist:
                ↪  "+str(dist)+"cm")
35
36      def exit(self):
37          self.obj_arduino.close_serial()
38
39  def main():
40      obj_led = LED_ON(115200)
41
42  if __name__== '__main__':
43      main()
```