**EEE 313 INTRODUCTION TO EMBEDDED SYSTEMS**

**FINAL EXAM APPLICATION HOMEWORKS**

**(25P) Q1.** Perform an application which must include the following elements with your Rpi, STM32 or Nano 33 board: At least (i) one LED (ii) one RGB LED (See below for the type), (iii) one potentiometer (except the one for LCD), (iv) one button, (v) one LCD, (v) one buzzer, (vi) one Motor, (vii) one Sensor (viii) one Sensor-2 (DHT or LDR for only the groups with 3 students). Note that you must use your assigned board, motor, sensor(s) and RGB LED. Also, you can use more than the required number and mentioned elements above, i.e, 2 LEDs, any resistor, any driver etc.

**Purpose of Application:** You are free about the purpose of application. Clearly state the intended purpose (i.e, what you are trying to do for each situation) in your video capture and in this document.

**Circuit Diagram:** You are free to build your circuit for application. Draw your circuit in **Fritzing.**

**Restriction:** Neither the Arduino IDE nor the Arduino programming language will be used when performing this application. Use only Thonny IDE with MicroPhython or STM32CubeIDE with C/C++.

**Homework Submission:** Record a video with all the team members for your application. In your video content; explain your program codes, show your program to be compiled successfully, show your program to be uploaded to your board, show your circuit to be run successfully for each case. Note that no simulation study is requested.

The following files need to be uploaded to Teams.

1. This word document by completing the ANSWERS section (do not upload as pdf)
2. Your video file (Will be talked in English)
3. Fritzing circuit file
4. Application project folder created by IDE software

------------------------------------------------ANSWERS-----------------------------------------------

**Project Team :** Suat Deniz – Ata Güneş

**Your Board :** Nvidia Jetson Nano Developer Kit

**Type of Your Motor :** Step Motor

**Your Sensor Type :** Motion Sensor

**Type of your RGB LED :** Common Cathode.

**Your Software IDE :** Gedit Text

**Your Programming Language :** Python

**Application Purpose :** The purpose of our project is to create an interactive and sensor-driven system using the Nvidia Jetson Nano Developer Kit, a step motor, a motion sensor, and a common cathode RGB LED. The project involves the integration of various electronic components, including an LED, a buzzer, and a button. When the button is pressed, the LED, buzzer, and RGB LED are activated simultaneously to provide a visual and auditory indication.

Additionally, the project incorporates a motion sensor to measure the proximity of objects. This information is displayed on an LCD screen in real-time, allowing users to monitor the distance detected by the motion sensor. The step motor is utilized to enhance the interactive experience by responding to the detected motion, creating a dynamic and responsive system.

Overall, our application aims to showcase the capabilities of the Nvidia Jetson Nano Developer Kit, demonstrate the integration of multiple electronic components, and provide a tangible example of sensor-based interaction through the combination of a motion sensor, step motor, RGB LED, and other peripheral devices.

**Fritzing Circuit Diagram:**

elektronik donanım, metin, elektronik mühendisliği, elektronik bileşen içeren bir resim

Açıklama otomatik olarak oluşturuldu

**Program codes:**

import Jetson.GPIO as GPIO

import time

trig\_pin = 8

echo\_pin = 7

servo\_pin = 12

GPIO.setmode(GPIO.BCM)

GPIO.setup(trig\_pin, GPIO.OUT)

GPIO.setup(echo\_pin, GPIO.IN)

GPIO.setup(servo\_pin, GPIO.OUT)

# Initialize servo

pwm = GPIO.PWM(servo\_pin, 50)

pwm.start(0)

# Initialize LCD

lcd\_rs = 23

lcd\_en = 24

lcd\_d4 = 5

lcd\_d5 = 6

lcd\_d6 = 13

lcd\_d7 = 19

lcd\_columns = 16

lcd\_rows = 2

LCD = [lcd\_rs, lcd\_en, lcd\_d4, lcd\_d5, lcd\_d6, lcd\_d7]

GPIO.setup(LCD, GPIO.OUT, initial=GPIO.LOW)

# Pin mapping for Jetson Nano Developer Kit

RS = 23

E = 24

D4 = 5

D5 = 6

D6 = 13

D7 = 19

# Initialize LCD

lcd\_rs = 23

lcd\_en = 24

lcd\_d4 = 5

lcd\_d5 = 6

lcd\_d6 = 13

lcd\_d7 = 19

# LCD constants

LCD\_WIDTH = 16

LCD\_HEIGHT = 2

def lcd\_init():

GPIO.setmode(GPIO.BCM)

GPIO.setup(led1\_pin, GPIO.OUT)

GPIO.setup(led2\_pin, GPIO.OUT)

GPIO.setup(RS, GPIO.OUT)

GPIO.setup(E, GPIO.OUT)

GPIO.setup(D4, GPIO.OUT)

GPIO.setup(D5, GPIO.OUT)

GPIO.setup(D6, GPIO.OUT)

GPIO.setup(D7, GPIO.OUT)

lcd\_function\_set()

lcd\_display\_control()

def lcd\_function\_set():

lcd\_byte(0x33, 0) # 110011 Initialize

lcd\_byte(0x32, 0) # 110010 Initialize

lcd\_byte(0x06, 0) # 000110 Cursor move direction

lcd\_byte(0x0C, 0) # 001100 Display On,Cursor Off, Blink Off

lcd\_byte(0x28, 0) # 101000 Data length, number of lines, font size

lcd\_byte(0x01, 0) # 000001 Clear display

time.sleep(0.0005) # Delay to allow commands to process

def lcd\_display\_control():

lcd\_byte(0x0C, 0) # 001100 Display On,Cursor Off, Blink Off

lcd\_byte(0x06, 0) # 000110 Cursor move direction

lcd\_byte(0x01, 0) # 000001 Clear display

time.sleep(0.0005) # Delay to allow commands to process

def lcd\_byte(bits, mode):

GPIO.output(RS, mode)

GPIO.output(D4, False)

GPIO.output(D5, False)

GPIO.output(D6, False)

GPIO.output(D7, False)

if bits & 0x10 == 0x10:

GPIO.output(D4, True)

if bits & 0x20 == 0x20:

GPIO.output(D5, True)

if bits & 0x40 == 0x40:

GPIO.output(D6, True)

if bits & 0x80 == 0x80:

GPIO.output(D7, True)

lcd\_toggle\_enable()

GPIO.output(D4, False)

GPIO.output(D5, False)

GPIO.output(D6, False)

GPIO.output(D7, False)

if bits & 0x01 == 0x01:

GPIO.output(D4, True)

if bits & 0x02 == 0x02:

GPIO.output(D5, True)

if bits & 0x04 == 0x04:

GPIO.output(D6, True)

if bits & 0x08 == 0x08:

GPIO.output(D7, True)

lcd\_toggle\_enable()

def lcd\_toggle\_enable():

time.sleep(0.0005) # Delay to allow commands to process

GPIO.output(E, True)

time.sleep(0.0005) # Delay to allow commands to process

GPIO.output(E, False)

time.sleep(0.0005) # Delay to allow commands to process

def lcd\_string(message, line):

if line == 0:

lcd\_byte(0x80, 0) # Set the cursor to the beginning of the first line

elif line == 1:

lcd\_byte(0xC0, 0) # Set the cursor to the beginning of the second line

else:

return # Invalid line number, do nothing

message = message.ljust(LCD\_WIDTH, " ")

for i in range(LCD\_WIDTH):

lcd\_byte(ord(message[i]), 1)

angle = 0

try:

while True:

GPIO.output(trig\_pin, GPIO.LOW)

time.sleep(0.2)

GPIO.output(trig\_pin, GPIO.HIGH)

time.sleep(0.00001)

GPIO.output(trig\_pin, GPIO.LOW)

while GPIO.input(echo\_pin) == 0:

pulse\_start = time.time()

while GPIO.input(echo\_pin) == 1:

pulse\_end = time.time()

pulse\_duration = pulse\_end - pulse\_start

distance = pulse\_duration \* 17150

distance = round(distance, 2)

lcd\_string("Distance: {} CM".format(distance), 0)

lcd\_string("Servo angle: {}d".format(angle), 1)

if distance <= 180:

angle = min(distance,180)

pwm.ChangeDutyCycle(2 + (angle / 18))

else:

angle = distance / 2

pwm.ChangeDutyCycle(2 + (angle / 18))

time.sleep(0.075)

except KeyboardInterrupt:

pass

finally:

    pass

**Photo for your circuit (only 1 photo):**

devre, Elektrik kabloları, Elektrik kaynağı, elektronik donanım içeren bir resim

Açıklama otomatik olarak oluşturuldu