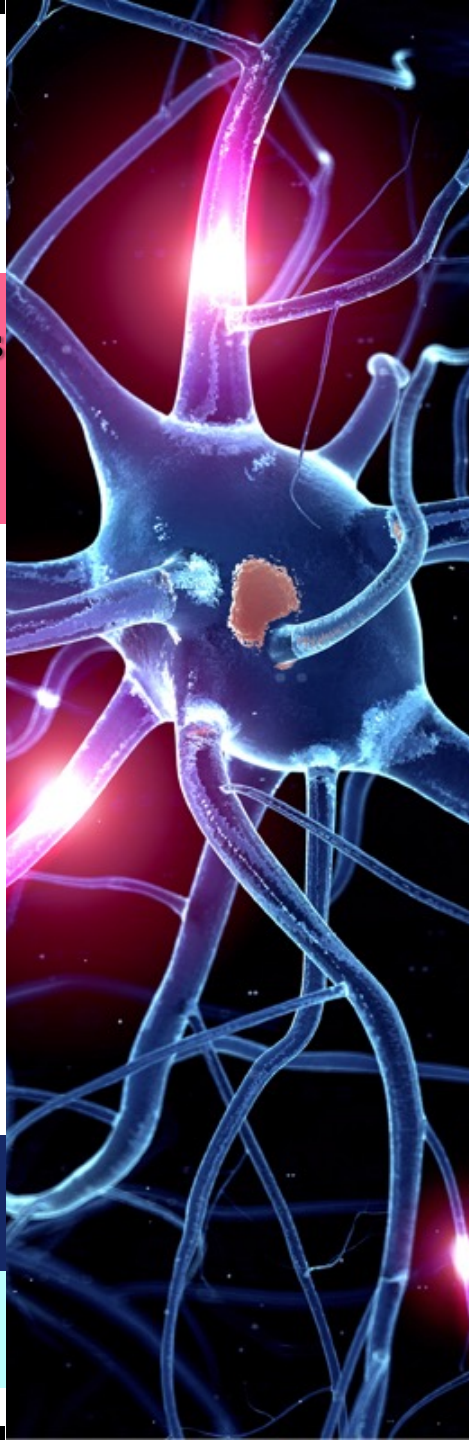
FUE
2022

Special Topics in Electronics and Communication Engineering - COM 581

ALS COMMUNICATION PROJECT



Under the Supervision of:
Dr. Ahmed Saeed

Ashraf Mahmoud *20184792*
Youssef Aser *20184117*

Date : 12 - 06 - 2022

CONTENTS

Contents

ACKNOWLEDGMENT

We would like to express our special thanks to Dr. Ahmed Saeed who gave us the opportunity to do this project (ALS Communication project), which helped us in doing a lot of Research and know how to use the ESP-32 Microprocessor, so we are thankful to you.
We had made this project with a lot of effort and utmost sincerity to complete it.

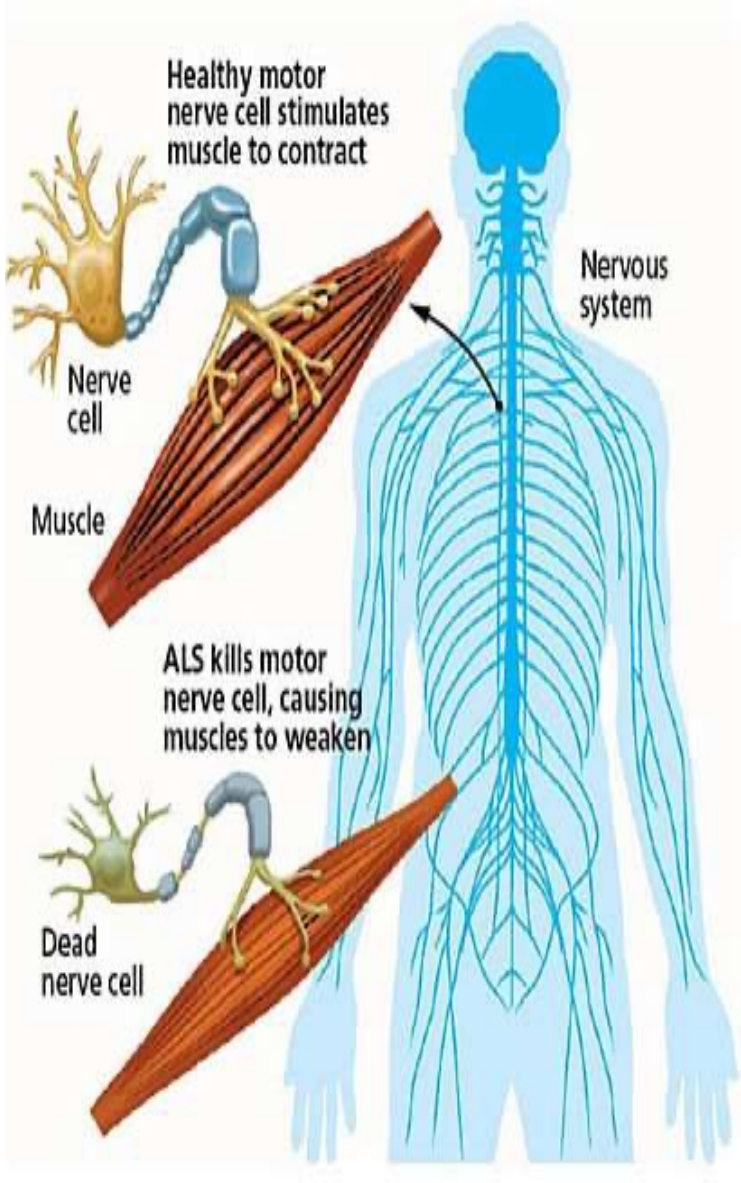
Acknowledgment

INTRODUCTION

- **Amyotrophic lateral sclerosis** or **ALS**, is a progressive nervous system disease that affects nerve cells in the brain and spinal cord, causing loss of muscle control.

- ALS often begins with muscle twitching and weakness in a limb, or slurred speech. Eventually, ALS affects control of the muscles needed to move, speak, eat and breathe. There is no cure for this fatal disease.

- So, we make this project to help ALS patients to communicate with people and to take information from his body like his/her heart rate and connect this information with his/her doctor to follow his/her case.

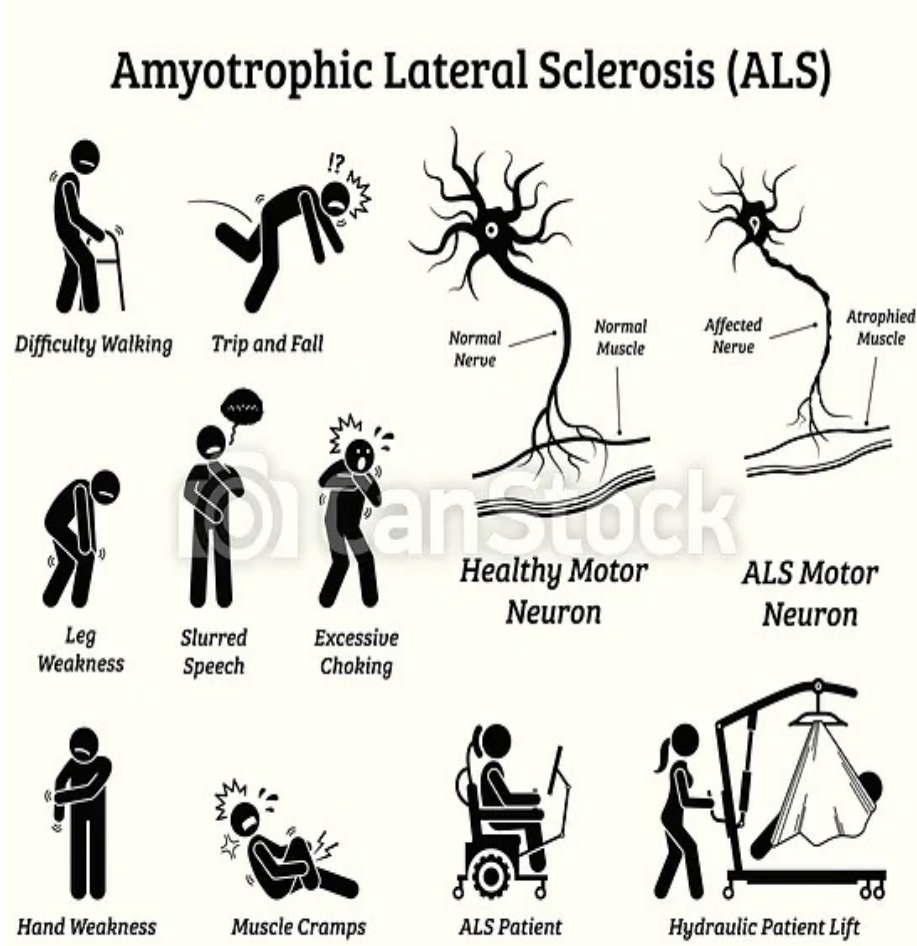


ALS DISEASE

Before we started our project, we did a lot of searches about ALS to study more about this disease and Its Symptoms and to know how can we help them, and we found that:

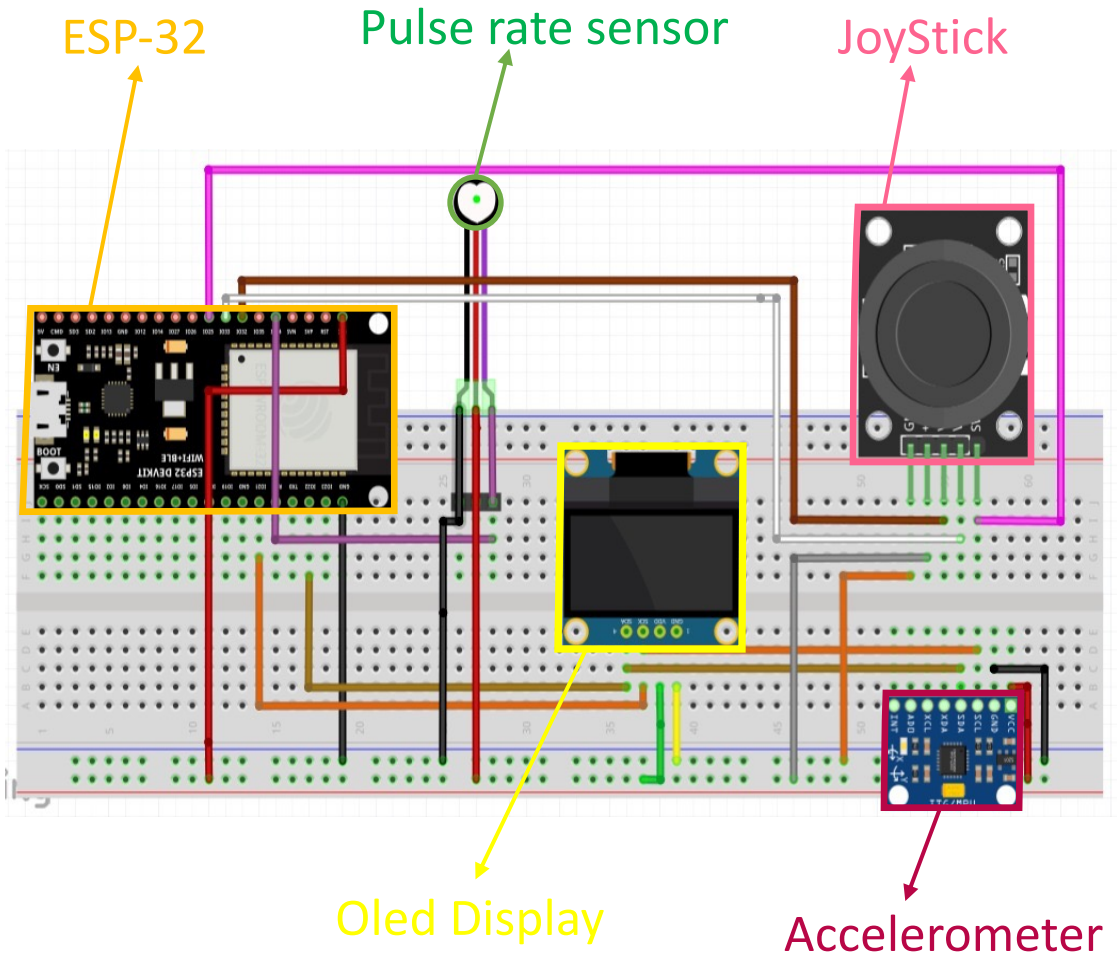
1. 80 to 95% of people with ALS are unable to meet their daily communication needs using natural speech.
2. Hand weakness or clumsiness.
3. Part of the nervous system that it affects governs cardiac function, including heart rate.
4. Inappropriate crying, laughing or yawning.
5. Cognitive and behavioral changes.
6. Difficulty walking or doing normal daily activities.
7. Weakness in legs, feet or ankles.[1]

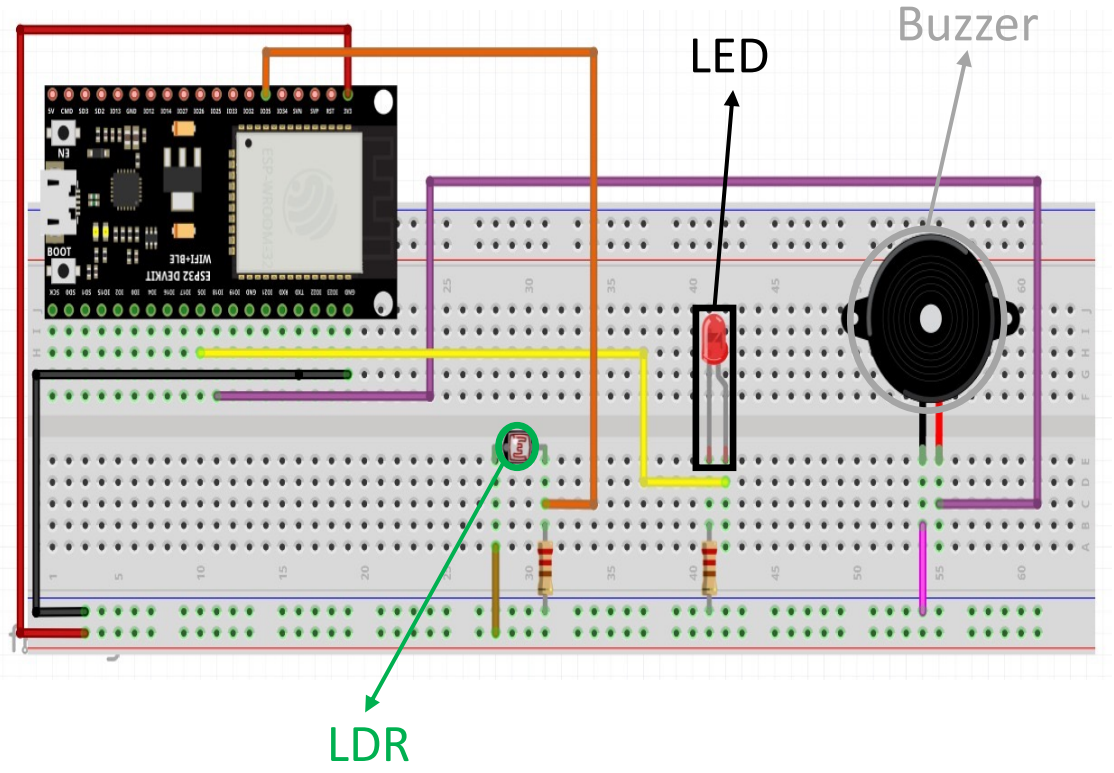
We take Stephen Hawking who was suffering from this disease as our case study.



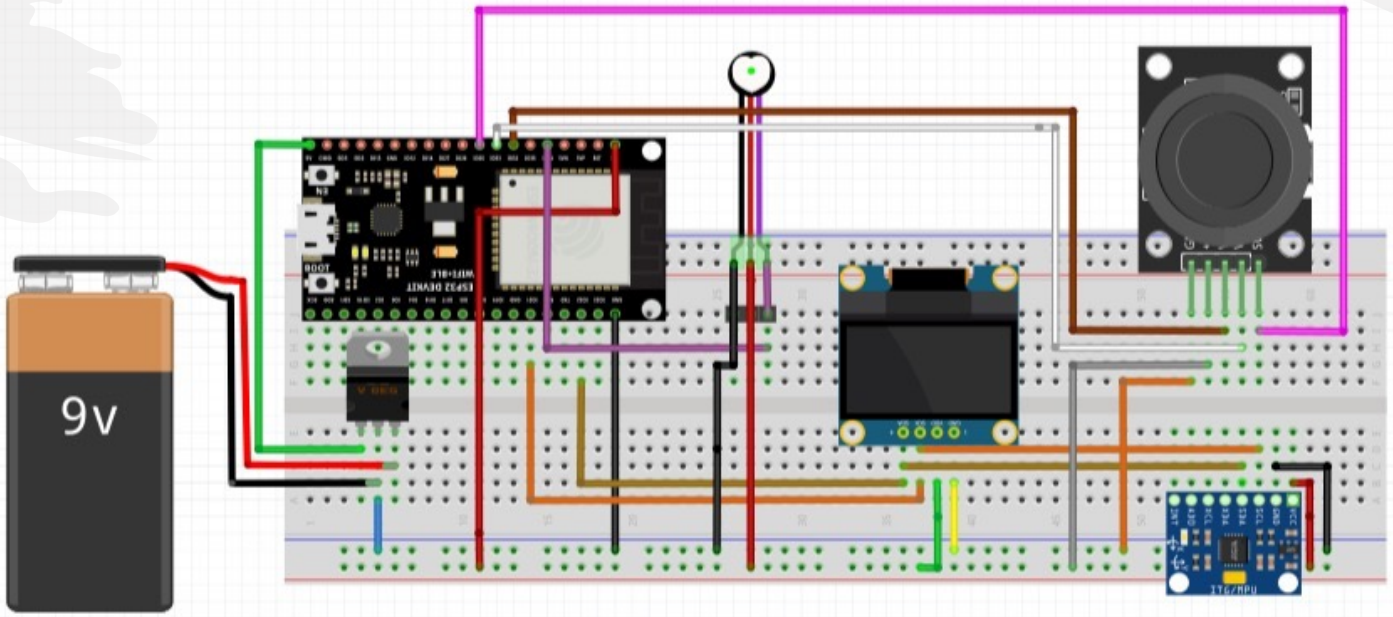
BLOCK DIAGRAM

fritzing



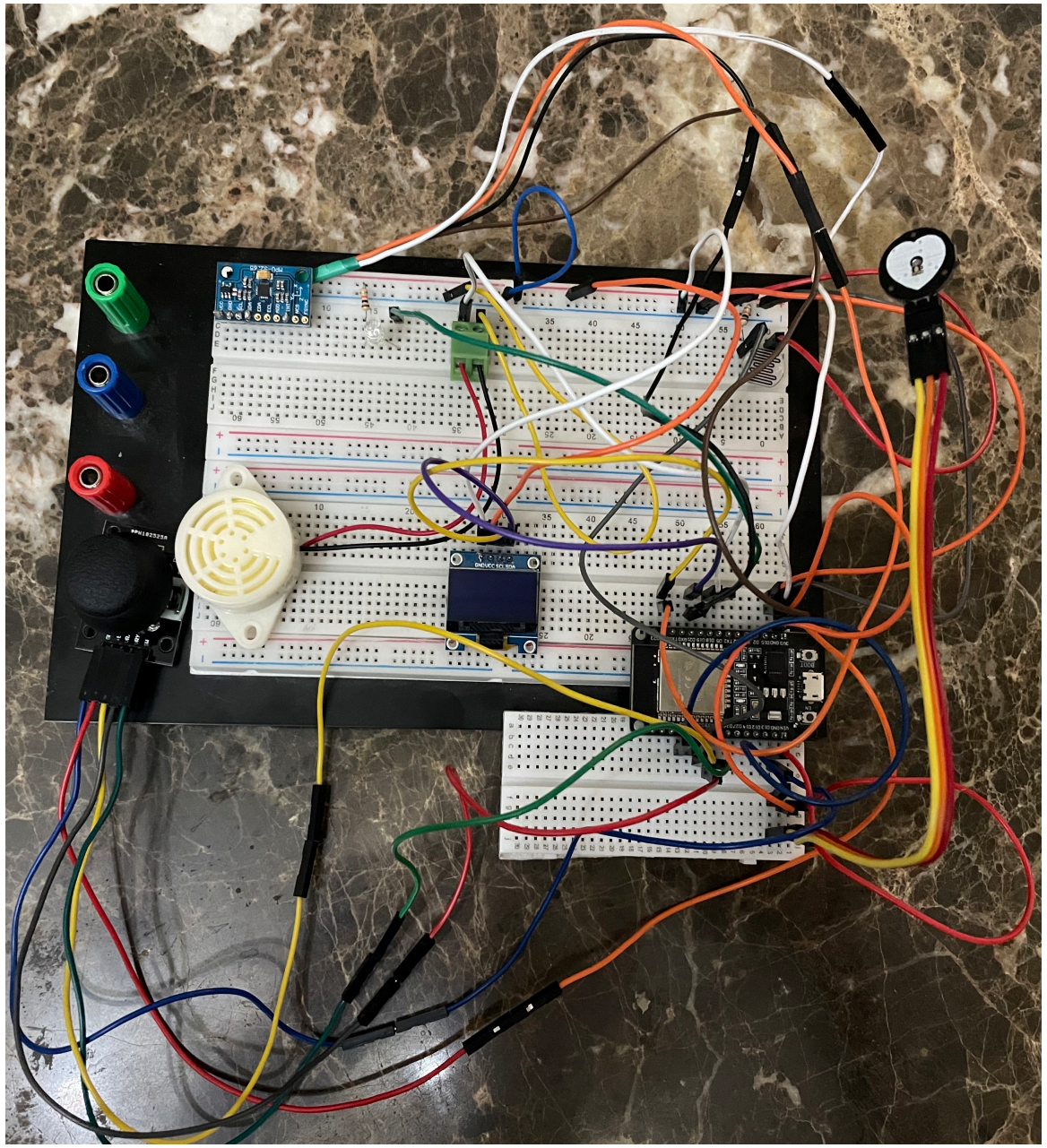


BLOCK DIAGRAM


9v
fritzing

DESIGN STEPS

- Connect +ve terminal of the LED to Pin 5 of ESP32 and –ve terminal by resistor to the GND.
- Connect terminal of the LDR to Vcc and the other terminal to Pin 35 of ESP32 and to the GND by resistor.
- Connect +ve terminal of the Buzzer to Pin 18 of ESP32 and –ve terminal to the GND.
- Connect Pulse Rate Sensor to the Vcc, GND and Pin 34.
- Connect Joystick using ADC Connection (Vcc, GND, VRx to Pin 32, VRy to Pin 33, SW to Pin 25).
- Connect Accelerometer using I2C Connection (Vcc, GND, SCL to Pin 22, SDA to Pin 21).
- Connect Oled display using I2C Connection too.



TESTING STRATEGY

Oled Display[4]

Thonny

ssd test.py

```
1  from machine import Pin, I2C
2  import ssd1306
3  from time import sleep
4
5
6  i2c = I2C(-1, scl=Pin(22), sda=Pin(21))
7
8  oled_width = 128
9  oled_height = 64
10 oled = ssd1306.SSD1306_I2C(oled_width, oled_height, i2c)
11
12 oled.text('Welcome to Lab!', 0, 0)
13 oled.text('Ashraf 20184792', 0, 16)
14 oled.text('Youssef 20184117', 0, 32)
15
16 oled.show()
```

Shell

```
>>> %Run -c $EDITOR_CONTENT
  Warning: I2C(-1, ...) is deprecated, use SoftI2C(...) instead
>>> %Run -c $EDITOR_CONTENT
  Warning: I2C(-1, ...) is deprecated, use SoftI2C(...) instead
>>> %Run -c $EDITOR_CONTENT
  Warning: I2C(-1, ...) is deprecated, use SoftI2C(...) instead
>>> %Run -c $EDITOR_CONTENT
  Warning: I2C(-1, ...) is deprecated, use SoftI2C(...) instead
>>>
```

LED, LDR & Buzzer

LDR+LED.py

```
1  from machine import Pin , ADC
2  from time import sleep
3
4  led = Pin(5 , Pin.OUT)
5  buzzer= Pin(18 , Pin.OUT)
6  ldr= ADC(Pin(35))
7  ldr.atten(ADC.ATTN_11DB)
8
9
10 while True:
11   ldr_value=ldr.read()
12   print('value = {}'.format(ldr_value))
13
14   if(ldr_value < 1000):
15       led.on()
16       buzzer.on()
17
18   else:
19       led.off()
20       buzzer.off()
21       sleep(0.5)
22
```

Shell

```
MicroPython v1.18 on 2022-01-17; ESP32 module with ESP32
Type "help()" for more information.
>>> %Run -c $EDITOR_CONTENT

  value = 2143
  value = 2106
  value = 2114
  value = 2094
  value = 634
  value = 615
  value = 605
  value = 576
  value = 550
  value = 547
  value = 527
```

Accelerometer[6]

ssd test.py | read joystick.py | main.py | Accel test.py

```
1  from machine import Pin, I2C ,ADC ,deepsleep
2  from time import sleep,time
3  from mpu import Accel
4  import ssd1306
5  import esp32
6
7  i2c = I2C(scl=Pin(22), sda=Pin(21))
8  mpu= Accel(i2c)
9
10 while True:
11     value = mpu.get_values()
12     y = value["AcY"]
13     x = value["AcX"]
14     z = value["AcZ"]
15
16     if y < 0 and x > 8000 :
17         print("right")
18
19     elif x < 0 and z > 0:
20         print("left")
21     elif y < 0 and x < 7000 and z < 7000:
22         print("back")
23     elif y > 7000 and x < 7000 and z < 7000:
24         print("forward")
25
26     else:
27         print("stop")
28
29     sleep(1.5)
30     print(value)
```

Shell

```
{'AcZ': 5940, 'AcY': -18152, 'AcX': 2796}
left
{'AcZ': 8392, 'AcY': 2540, 'AcX': -13716}
forward
{'AcZ': -496, 'AcY': 9632, 'AcX': -13168}
stop
{'AcZ': -652, 'AcY': 18348, 'AcX': 19768}
right
{'AcZ': 420, 'AcY': -11368, 'AcX': 15040}
back

MicroPython v1.18 on 2022-01-17; ESP32 module with
Type "help()" for more information.
>>>
```

Joystick[5]

Thonny - /

ssd test.py | read joystick.py

```
1  from machine import Pin, ADC
2  from time import sleep_ms
3
4  x = ADC(Pin(32, Pin.IN))
5  y = ADC(Pin(33, Pin.IN))
6  x.atten(ADC.ATTN_11DB)
7  y.atten(ADC.ATTN_11DB)
8
9  while True:
10     x_val = x.read()
11     y_val = y.read()
12     print('Current position:{},{}'.format(x_val,y_val))
13     sleep_ms(300)
```

Shell

```
Current position:1835,4095
Current position:1837,1943
Current position:0,1947
Current position:1831,1947
Current position:1834,0
Current position:1839,1949
Current position:1840,1949
Current position:1837,1951
Current position:1839,1950
Current position:1835,1950

MicroPython v1.18 on 2022-01-17; ESP32 module with ESP32
Type "help()" for more information.
>>>
```

Pulse Rate Sensor[2]

Thonny - /Users/ashrafmahn

heart sensor2.py

```
from machine import Pin, Signal, ADC, Timer
adc = ADC(Pin(34))

# On my board on = off, need to reverse.
led = Signal(Pin(2, Pin.OUT), invert=True)

MAX_HISTORY = 250

# Maintain a log of previous values to
# determine min, max and threshold.
history = []
beat = False
beats = 0

def calculate_bpm(t):
    global beats
    print('BPM:', beats * 6) # Triggered every 10 seconds, * 6 = bpm
    beats = 0

timer = Timer(1)
timer.init(period=10000, mode=Timer.PERIODIC, callback=calculate_bpm)

while True:
    v = adc.read()

    history.append(v)

    # Get the tail, up to MAX_HISTORY length
    history = history[-MAX_HISTORY:]

    minima, maxima = min(history), max(history)
```

Shell

```
MicroPython v1.18 on 2022-01-17; ESP32 module with ESP32
Type "help()" for more information.
>>> %Run -c $EDITOR_CONTENT

  BPM: 0
  BPM: 54
  BPM: 42
  BPM: 48

MicroPython v1.18 on 2022-01-17; ESP32 module with ESP32
Type "help()" for more information.
>>>
```

WIFI & Thing Speak

Thonny - /Users/ashrafmahmoud/Downloads/wifi+TS TEST.py @ 38 : 29

Final.py | wifi+TS TEST.py *

```
import machine
import urequests
from machine import Pin, SoftI2C , ADC
import network, time

from time import sleep

HTTP_HEADERS = {'Content-Type': 'application/json'}
THINGSPEAK_WRITE_API_KEY = '7UKLHPE2R62DOTXJ'

UPDATE_TIME_INTERVAL = 5000  # in ms
last_update = time.ticks_ms()

ssid='AM'
password='941999*#'

# Configure ESP32 as Station
sta_if=network.WLAN(network.STA_IF)
sta_if.active(True)

if not sta_if.isconnected():
    print('connecting to network...')
    sta_if.connect(ssid, password)
    while not sta_if.isconnected():
     pass
print('network config:', sta_if.ifconfig())

while True:
    if time.ticks_ms() - last_update >= UPDATE_TIME_INTERVAL:

        ldr=ADC(Pin(35))
        ldr.atten(ADC.ATTN_11DB)
        x=ldr.read()

        bme_readings = {'field1':x}
        request = urequests.post( 'http://api.thingspeak.com/update?api_key=' + THINGSPEAK_WRITE_API_KEY, json = bme_readings, headers = HTTP_HEADERS )
        request.close()
        print(bme_readings)
```

Shell

```
connecting to network...
network config: ('192.168.1.10', '255.255.255.0', '192.168.1.1', '217.52.47.130')
{'field1': 2774}
{'field1': 2771}
{'field1': 2775}
{'field1': 2773}
{'field1': 2773}
```

Overall System

Final.py

```
from machine import Pin, I2C ,ADC
from time import sleep
from mpu import Accel
import ssd1306
import urequests
import network, time

#For the led , buzzer & LDR
led = Pin(5 , Pin.OUT)
buzzer= Pin(18 , Pin.OUT)
ldr= ADC(Pin(35))
ldr.atten(ADC.ATTN_11DB)

#For the joystick
X = ADC(Pin(32, Pin.IN))
Y = ADC(Pin(33, Pin.IN))
X.atten(ADC.ATTN_11DB)
Y.atten(ADC.ATTN_11DB)

#For the dislpay &accelometer
i2c = I2C(scl=Pin(22), sda=Pin(21))
mpu= Accel(i2c)

oled_width = 128
oled_height = 64
oled = ssd1306.SSD1306_I2C(oled_width, oled_height, i2c)

#For ThingSpeak
HTTP_HEADERS = {'Content-Type': 'application/json'}
THINGSPEAK_WRITE_API_KEY = '7UKLHPE2R62DOTXJ'

UPDATE_TIME_INTERVAL = 5000  # in ms
last_update = time.ticks_ms()

ssid='AM'
password='941999*#'

# Configure ESP32 as Station
sta_if=network.WLAN(network.STA_IF)
```

Shell

```
{'AcZ': 1280, 'AcY': -16032, 'AcX': -356}
stop
left
{'AcZ': 1300, 'AcY': -15972, 'AcX': -356}
stop
left
{'AcZ': 1328, 'AcY': -15952, 'AcX': -372}
```

Final.py

```
# Configure ESP32 as Station
sta_if=network.WLAN(network.STA_IF)
sta_if.active(True)

if not sta_if.isconnected():
    print('connecting to network...')
    sta_if.connect(ssid, password)
    while not sta_if.isconnected():
     pass
print('network config:', sta_if.ifconfig())


while True:


    ldr_value=ldr.read()

    value = mpu.get_values()
    y = value["AcY"]
    x = value["AcX"]
    z = value["AcZ"]

    oled.invert(True)

    if(ldr_value < 1000):
        led.on()
        buzzer.on()

    else:
        led.off()
        buzzer.off()
    sleep(0.5)

    if y < 0 and x > 8000 :
        print("right")
        oled.fill(0)
        oled.text('Eat', 50, 32)
```

Final.py

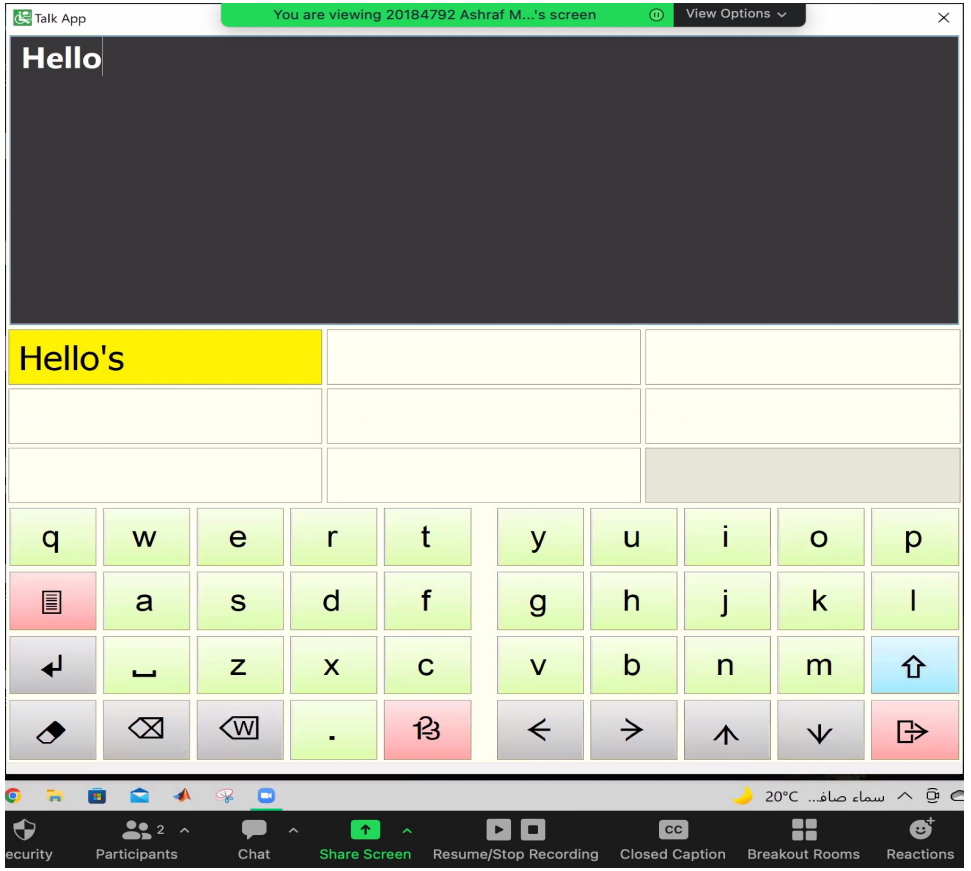
```
77
78      elif x < 0 and z > 0:
79          print("left")
80          oled.fill(0)
81          oled.text('Sleep', 50, 32)
82          oled.show()
83      elif y < 0 and x < 7000 and z < 7000:
84          print("back")
85          oled.fill(0)
86          oled.text('toilet', 50, 32)
87          oled.show()
88      elif y > 7000 and x < 7000 and z < 7000:
89          print("forward")
90          oled.fill(0)
91          oled.text('tired', 50, 32)
92          oled.show()
93          buzzer.on()
94
95      else:
96          oled.fill(0)
97          print("stop")
98
99
100     print(value)
101
102     if time.ticks_ms() - last_update >= UPDATE_TIME_INTERVAL:
103
104
105         x=ldr.read()
106
107         ldr_readings = {'field1':x}
108         request = urequests.post( 'http://api.thingspeak.com/update?api_key=' + THINGSPEAK_WRITE_API_KEY, json = ldr_readings, headers = HTTP_HEADERS )
109         request.close()
110         print(ldr_readings)
111
112
113
114
115     while True:
```

Final.py

```
114
115     while True:
116
117         xx = X.read()
118         yy = Y.read()
119
120         if yy==4095 and 4095 > xx  :
121             print("right")
122             oled.fill(0)
123             oled.text('Yes', 50, 32)
124             oled.show()
125
126
127         elif  xx < 4095 and yy==0 :
128             print("left")
129             oled.fill(0)
130             oled.text('No', 50, 32)
131             oled.show()
132
133         elif  yy < 4095 and xx==0 :
134             print("back")
135             oled.fill(0)
136             oled.text('IDK', 50, 32)
137             oled.show()
138
139         elif xx==4095 and 4095 > yy:
140             print("forward")
141             oled.fill(0)
142             oled.text('Thanks', 50, 32)
143             oled.show()
144
145         else:
146             print("stop")
147             oled.fill(0)
148
149
150         break
151     sleep(0.2)
```

CONCLUSION

We do our best to try to help ALS patients with our limited Possibilities and components, and we are sure that our project can be improved using advanced components like Brain Signal Sensors and use programs like ACAT which can make an interface from human interactions to the device[3], and we tried to use it in our project, but we couldn’t finish it on time.



REFERENCES

References

[1] M. C. Staff, "MAYO CLINIC," 22 2 2022. [Online]. Available: https://www.mayoclinic.org/diseases-conditions/amyotrophic-lateral-sclerosis/symptoms-causes/syc-20354022.

[2] M. Fitzpatrick, "mfitz," 21 1 2021. [Online]. Available: https://www.mfitzp.com/wemos-heart-rate-sensor-display-micropython/.

[3] "Intel," [Online]. Available: https://www.intel.com/content/www/us/en/support/articles/000015001/programs.html.

[4] "Microcontrollers Lab," [Online]. Available: https://microcontrollerslab.com/micropython-oled-display-esp32-esp8266/.

[5] "TechTOThinker," [Online]. Available: https://techtotinker.blogspot.com/2021/04/025-micropython-technotes-joystick.html.

[6] "Github," [Online]. Available: https://github.com/topics/mpu9250.