BreadBoard Guitar

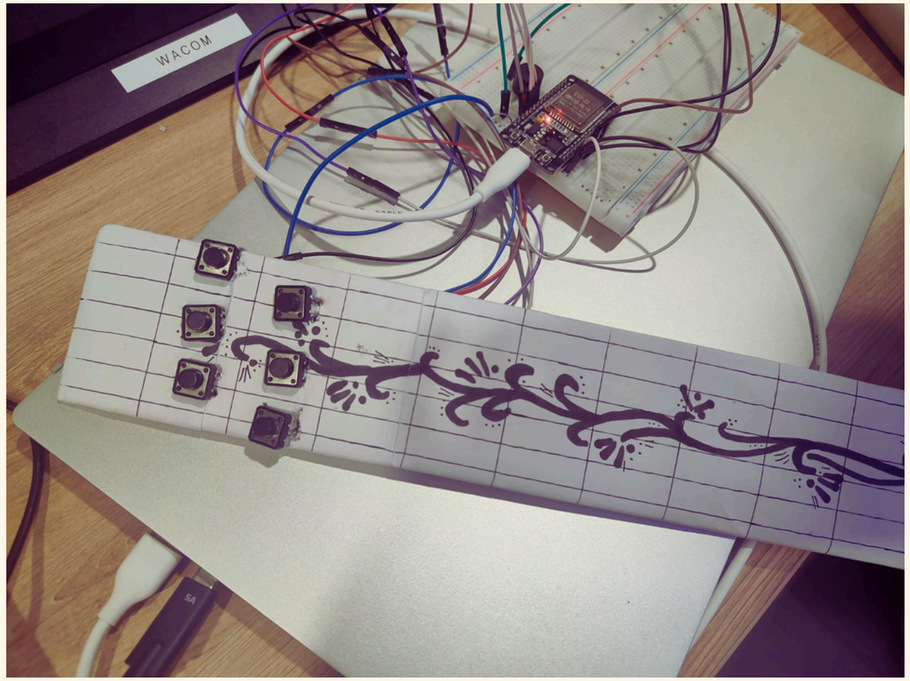
ur fav team btw

101

Punya & Zalak

the idea was to create a toy guitar that mimics the 3 chords : D, A, and G through the pushbuttons on the fretboard.

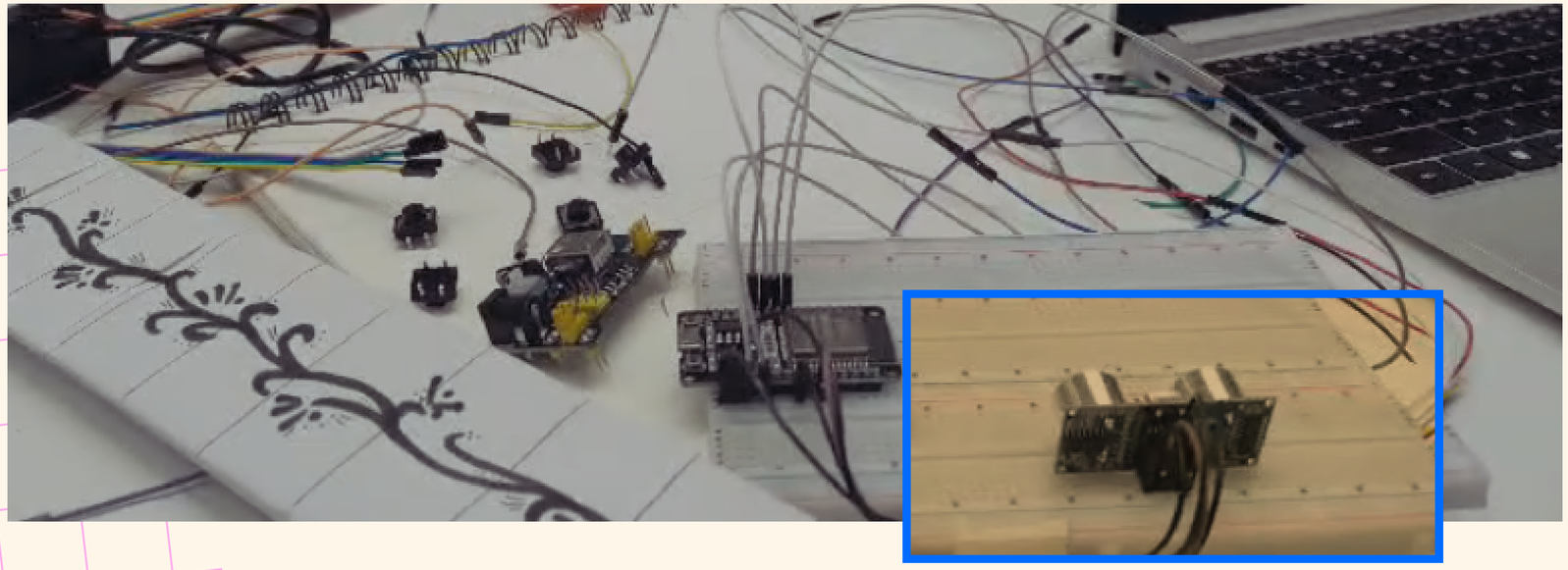
the idea



#fire

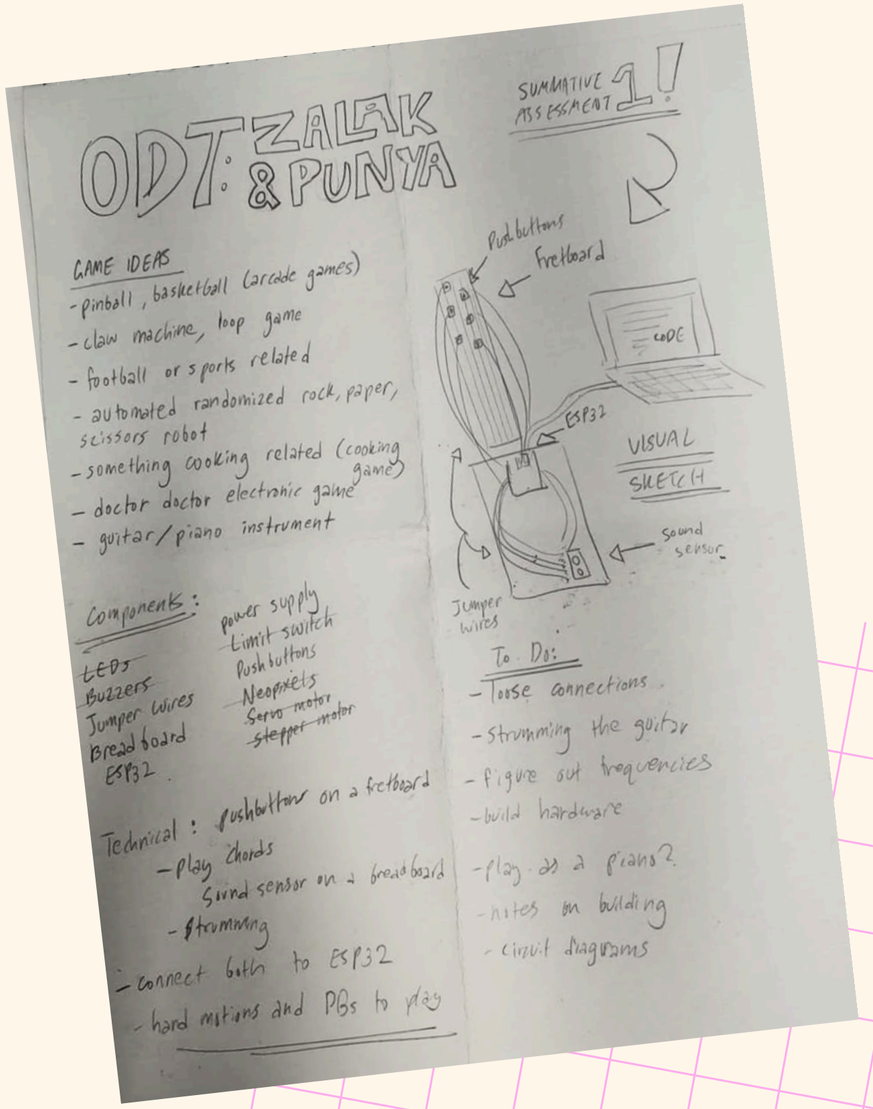
to make it even more closer to an actual guitar, I decided to introduce the ultrasonic sensor for the strumming.

the idea but it gets better

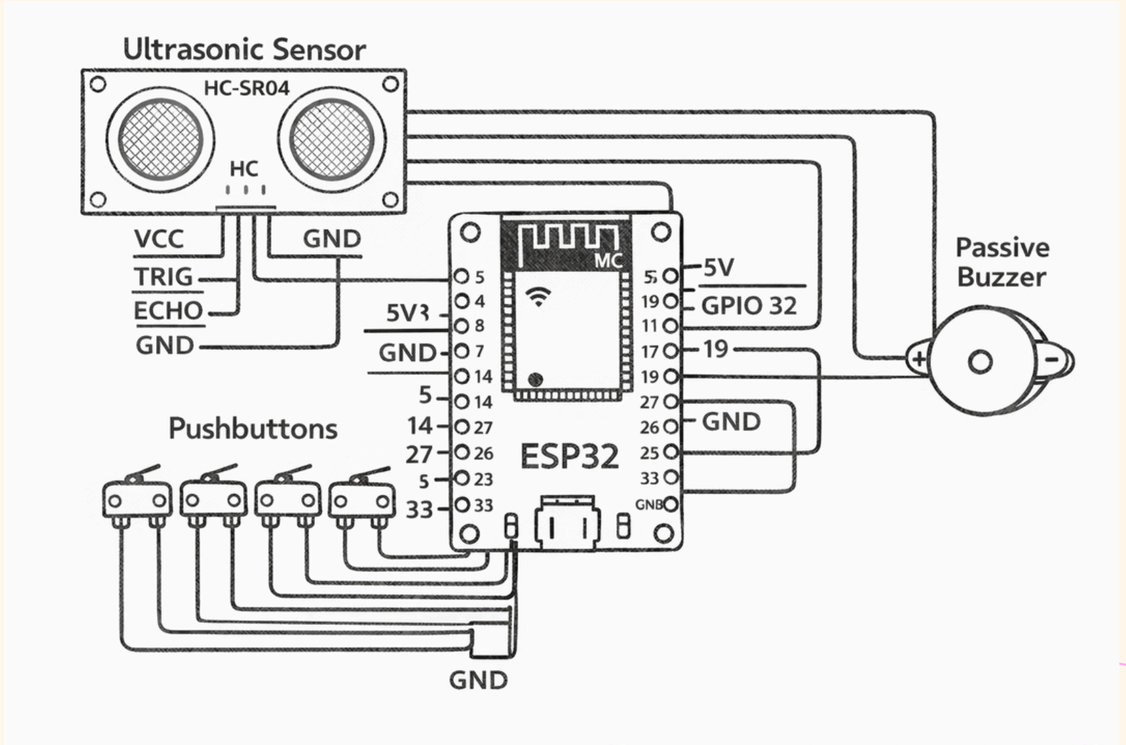


#evenfirer

IDEATION


ODT: ZALAK & PUNYA
SUMMATIVE ASSESSMENT 1!
GAME IDEAS
- pinball, basketball (arcade games)
- claw machine, loop game
- football or sports related
- automated randomized rock, paper, scissors robot
- something cooking related (cooking game)
- doctor doctor electronic game
- guitar/piano instrument
Components:
~~LEDs~~
~~Buzzers~~
Jumper wires
Bread board
ESP32
power supply
Limit switch
Push buttons
Neopixels
~~Servo motor~~
~~Stepper motor~~
Technical: pushbuttons on a fretboard
- play chords
Sound sensor on a breadboard
- strumming
- connect both to ESP32
- hard motions and PBs to play
Push buttons
Fretboard
CODE
ESP32
VISUAL SKETCH
Sound sensor
Jumper wires
To Do:
- Loose connections
- strumming the guitar
- figure out frequencies
- build hardware
- play as a piano?
- notes on building
- circuit diagrams

CIRCUIT DIAGRAM



AI GENERATED

the big bad code

```python
from machine import Pin, PWM, time_pulse_us
import time

audio = PWM(Pin(32))
audio.duty(0)

s1= Pin(15,Pin.IN,Pin.PULL_UP)
s2= Pin(4,Pin.IN,Pin.PULL_UP)
s3= Pin(14,Pin.IN,Pin.PULL_UP)
s4= Pin(27,Pin.IN,Pin.PULL_UP)
s5= Pin(26,Pin.IN,Pin.PULL_UP)
s6= Pin(25,Pin.IN,Pin.PULL_UP)

trig= Pin(5,Pin.OUT)
echo= Pin(19,Pin.IN)

D_chord = [294, 370, 440]
A_chord = [220, 277, 330]
G_chord = [196, 247, 294]

def get_distance():
    trig.off()
    time.sleep_us(2)
    trig.on()
    time.sleep_us(10)
    trig.off()

    duration = time_pulse_us(echo, 1, 30000)

    if duration < 0:
        return None

    distance = duration / 58
    return distance
```

```python
def play_down(chord):
    for note in chord:
        audio.freq(note)
        audio.duty(500)
        time.sleep(0.08)
        audio.duty(0)
        time.sleep(0.02)

def play_up(chord):
    for note in reversed(chord):
        audio.freq(note)
        audio.duty(500)
        time.sleep(0.08)
        audio.duty(0)
        time.sleep(0.02)


last_distance = get_distance()
current_chord = None

while True:

    c1= s1.value()
    c2= s2.value()
    c3= s3.value()
    c4= s4.value()
    c5= s5.value()
    c6= s6.value()

    if c1==0 and c2==0 and c3==0:
        current_chord= D_chord
    elif c1==0 and c5==0 and c6==0:
        current_chord= G_chord
    elif c5==0 and c4==0 and c3==0:
        current_chord= A_chord

    else:
        current_chord= None
        audio.duty(0)
```

```
distance = get_distance()

if distance is None:
    continue



change = distance - last_distance

if current_chord != None:
    audio.duty(0)

    if change < -6:
        play_down(current_chord)

    if change > 6:
        play_up(current_chord)


print("distance:", distance, " Change:", change)

last_distance = distance
time.sleep(0.05)
```

the big bad code

coding shoding

MY CONTRIBUTION

software development

- Wrote the complete MicroPython code for the ESP32.
- Programmed ultrasonic distance measurement using the HC-SR04.
- Implemented distance change detection to simulate guitar strumming.
- Created logic to detect up-strum and down-strum based on movement direction.
- Designed chord mappings using frequency arrays (D, A, G).
- Controlled a passive buzzer using PWM signals.
- Added fade-in and fade-out effects to reduce robotic sound.
- Debugged constant buzzing and undefined variable errors.
- Optimized timing and thresholds for smoother performance.

hardware configuration

- Connected and configured the ESP32 with:
  - Ultrasonic sensor (TRIG & ECHO pins)
  - Pushbuttons for chord selection
  - Passive buzzer for sound output
- Used GPIO pins efficiently for multiple inputs.
- Configured internal pull-up resistors for stable button readings.
- Ensured proper grounding and power distribution.
- Tested and corrected wiring issues.

The entire project returns to one central purpose: to celebrate the joy of building something wonderfully unnecessary. The breadboard guitar is not positioned as a revolutionary instrument or an essential tool.

It exists simply to spark curiosity, laughter, and experimentation.


thanks