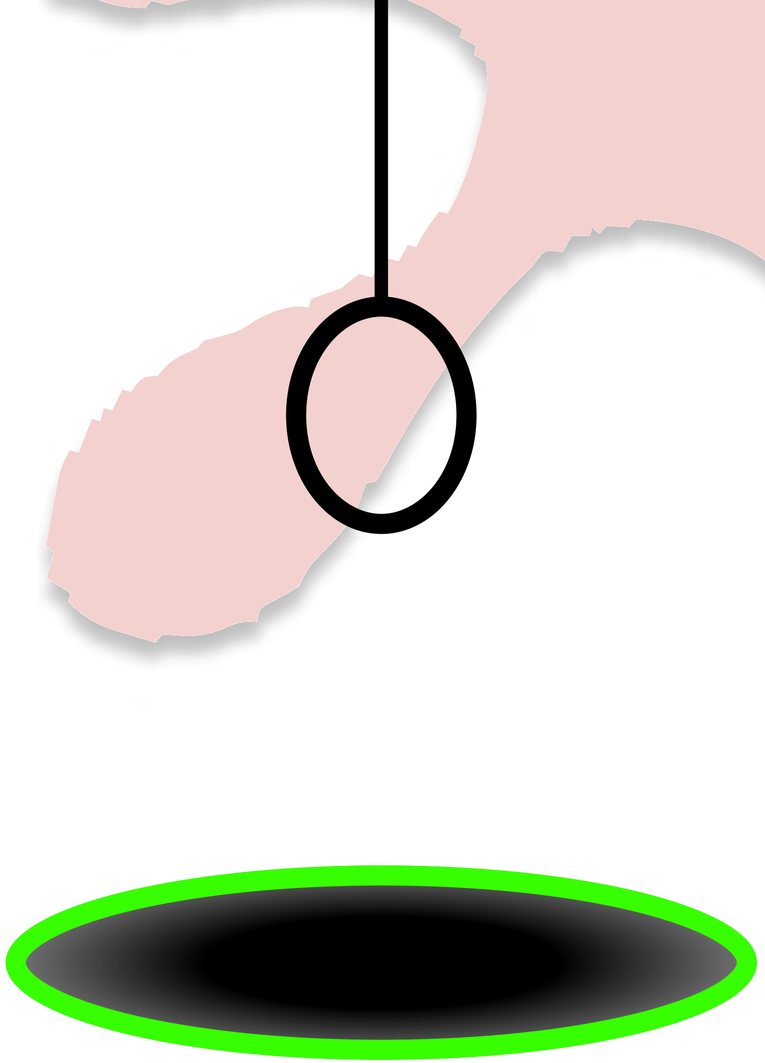
HANGMAN

Morse Code Edition

.-.-.-.---.-.----.



BY Shantanu and Avyukt

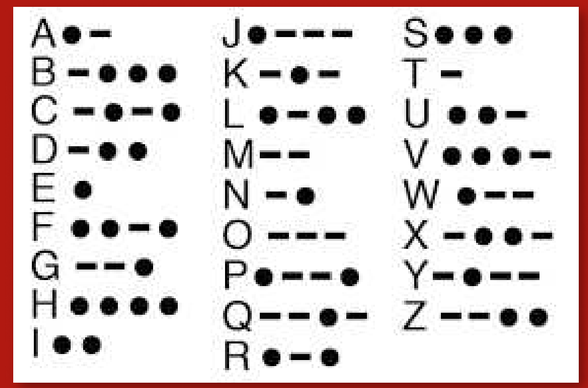
How'd we come to this idea?

We had multiple iterations, like

1. memory game
2. basketball
3. talking robot

But these were already made, and we wanted to challenge ourselves with code...
So we decided to build hangman but with MULTIPLE states

BUT HOW DO WE MAKE IT ABSURD?????
WHY NOT ADD MORSE CODE TO IT!!!



Game Concept

Once Morse code was locked in, the next question was:

How do we gamify this?

1. A random word is selected.
2. The player enters letters using Morse code, entered through short and long presses
3. Each decoded letter is checked against the word.
4. Correct guesses reveal letters, while wrong guesses count as mistakes.
5. The player has a limited number of mistakes.
6. Visual, sound, and physical feedback indicate progress.
7. Guess the full word to win; exceed the mistake limit to lose.

Components used?

Push Button (-1)

Role: Primary user input

Function:

- used to enter Morse code
- Short press = dot (.)
- Long press = dash (-)
- Press duration is measured to convert time into data

Buzzers (-1)

Role: Immediate auditory feedback

Function:

- Turns ON while the button is pressed
- Helps the user judge press length without looking
- Makes the interaction tactile and instinctive

NeoPixel (16 LEDs)

Role: Visual feedback and game progress indicator

Function:

- Divided into segments per guessed letter
- Green LEDs → correct guess
- Red LEDs → wrong guess
- Full-strip animations for win or loss

LEDs (2)

Role: Low-level real-time feedback

Function:

- Light up during Morse input
- Show whether the last signal was a dot or dash (ONE led lights up for dot, TWO for dash)
- Helps during learning and debugging

Servo Motor (1)

Role: Physical consequence indicator

Function:

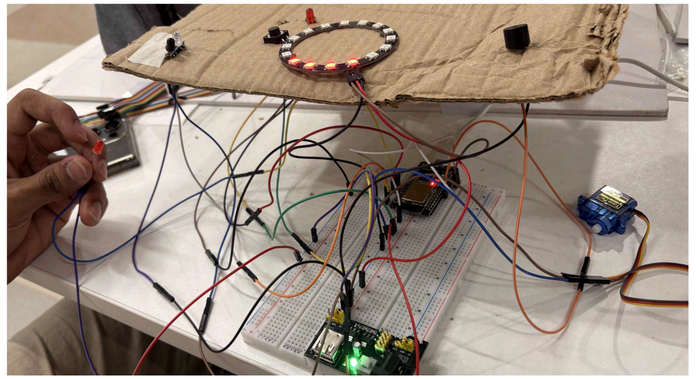
- Moves 90° when the user fails making the kid fall into the hole
- Resets to position every new game

IR Sensor (1)

Role: Game reset controller

Function:

- Instantly resets the game state
- Clears LEDs
- Picks a new random word
- Allows replay without power cycling



Coding Logic?

State Machines

- Pressed vs released
- In-game vs game-over
- Entering letter vs waiting

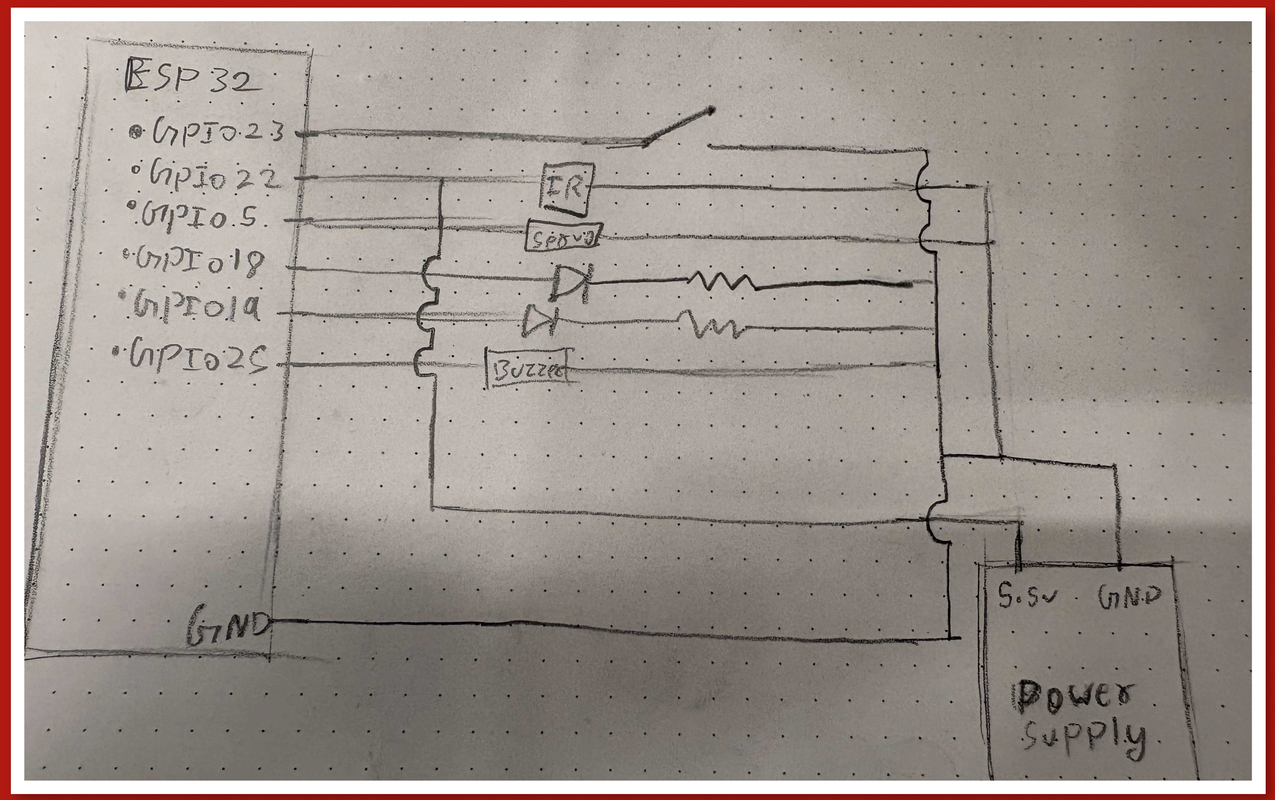
Time Thresholds

- Short press → dot
- Long press → dash
- Long pause → letter complete

Fail Logic

- Invalid Morse → mistake
- 3 mistakes → game ends
- No soft recovery

Circuit diagram



Pain Points :(

Can an ESP32 accurately measure press durations? BUT WE DID SOLVE THIS

Can timing gaps be reliably detected? WE SOLVED THIS TOO

Can a single button reliably detect precise Morse input without debounce errors?

NAYAN please help us with this

Learnings!!

1. Simple systems still need safety logic
   Even with one button, edge cases must be anticipated and handled explicitly.

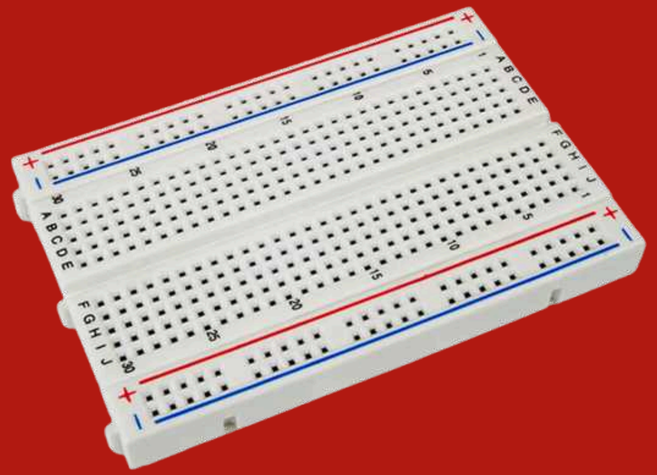
2. Thresholds need a loott testing and can't be guesswork
   Timing constants (dot/dash threshold, letter gap) worked only after repeated physical testing, not randomly guessed values. We spent half an hour getting this right...

3. Know Your Code
   Even though our code worked, our ppt went great, but we weren't able to explain a concept... why? We didn't understand what our code was even if it worked.

Individual Contribution

Shantanu?

1. 60% contribution in the Coding
2. 40% contribution in building of the hardware
3. 50% contribution in building of the game model



Avyukt

1. 40% contribution in the Coding
2. 60% contribution in building of the hardware
3. 50% contribution in building of the game model

Thank you

From Shantanu and Avyukt