Analog Input & Output

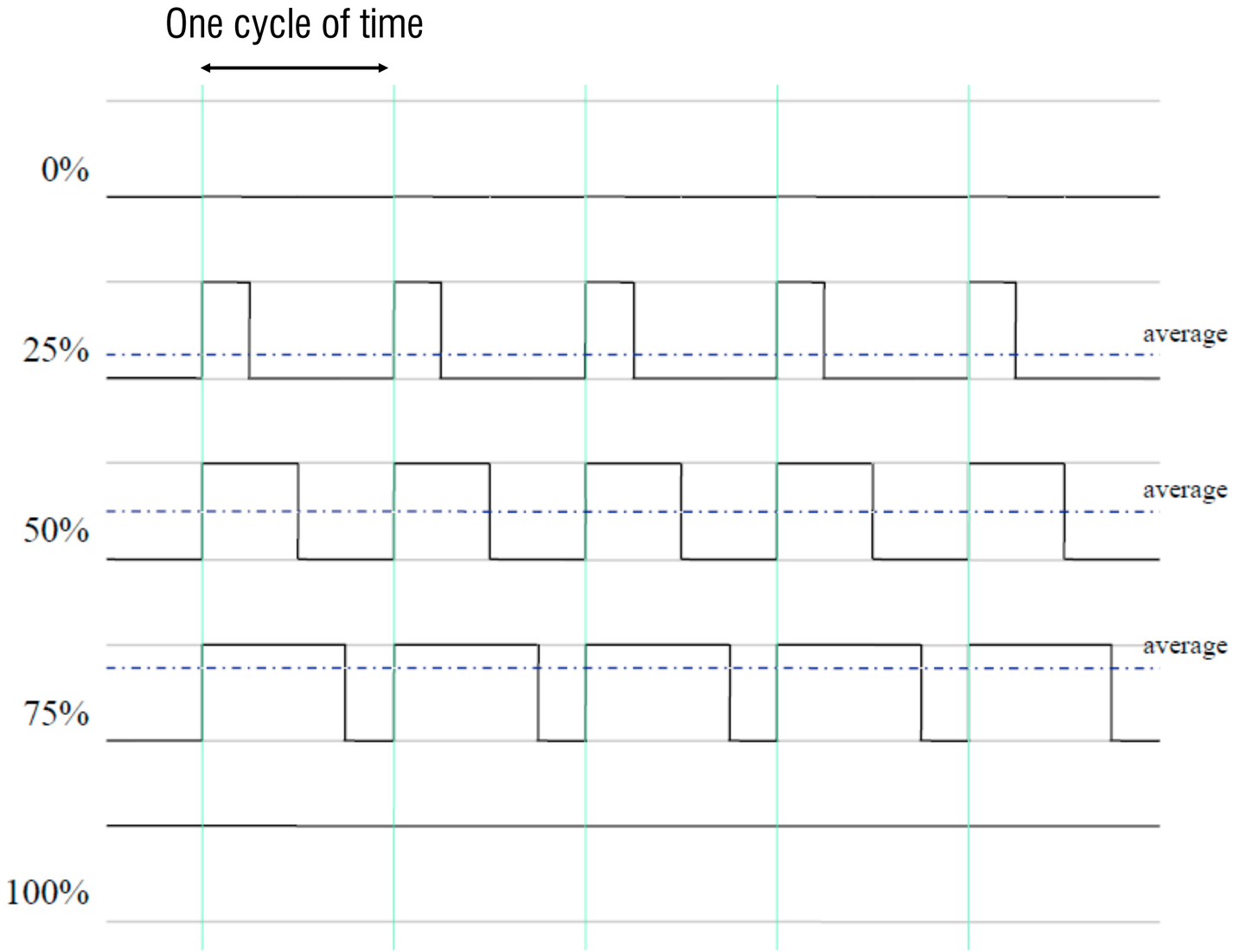
Huaishu Peng | UMD CS | Fall 2024

Pulse Width Modulation (PWM)

A technique for simulating **analog** results with **digital** signal

Pausing the power supply **ON** and **OFF** at a certain **frequency**
And with a certain pause **width**

It allows us to control the light intensity, speed of the motor etc.



analogWrite() is on a scale of 0 (always off) – 255 (always on)
Pins that support PWM: all GPIO pins except 6-11 and 34-39

Mini program 1: Breathing effect

1. Find the Red LED
2. Create a fading/breathing effect - change the LED's light intensity with analogWrite()

analogWrite() is on a scale of 0 (always off) – 255 (always on)
Pins that support PWM: **all GPIO pins except 6-11 and 34-39**

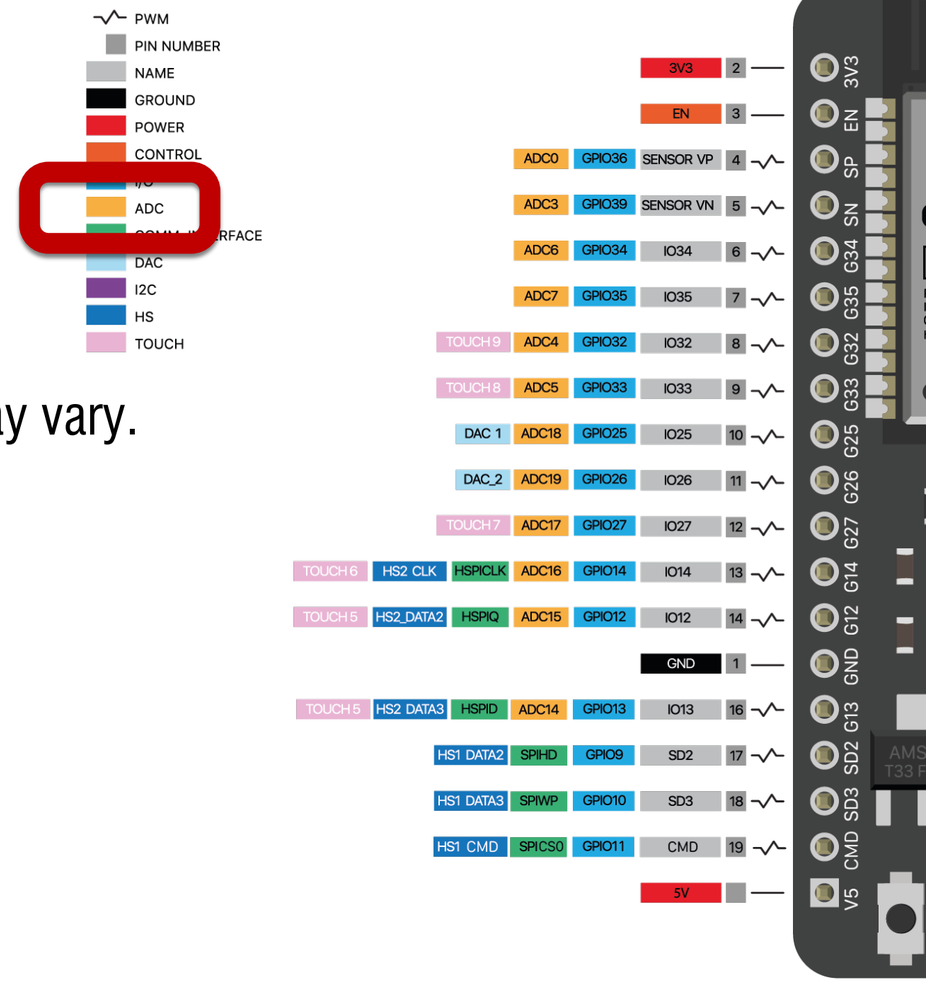
Hint:

- Use GPIO23 to control LED

Analog Input

Analog Input (Analog-to-Digital Converter or ADC)

you can measure varying voltage levels between 0 V and 3.3 V – Provide us a richer understanding of the environment.



Arduino functions

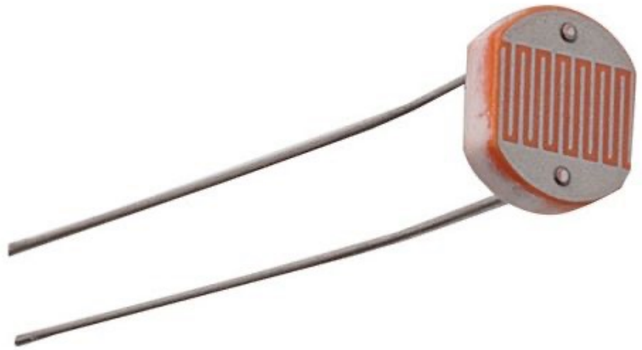
– int **analogRead(pin)** to read the voltage value of a pin

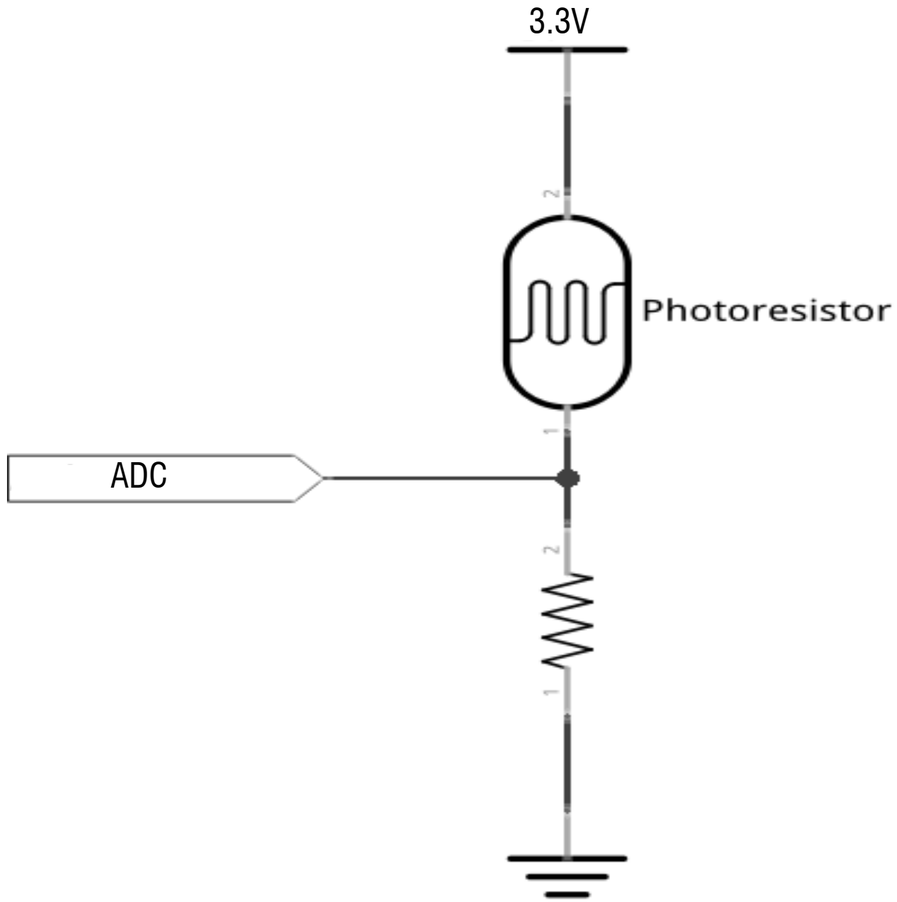
– Depending on the board you use, the Analog Pin and it's resolution may vary.

– For the ESP32: we can use up to 18 ADC channels

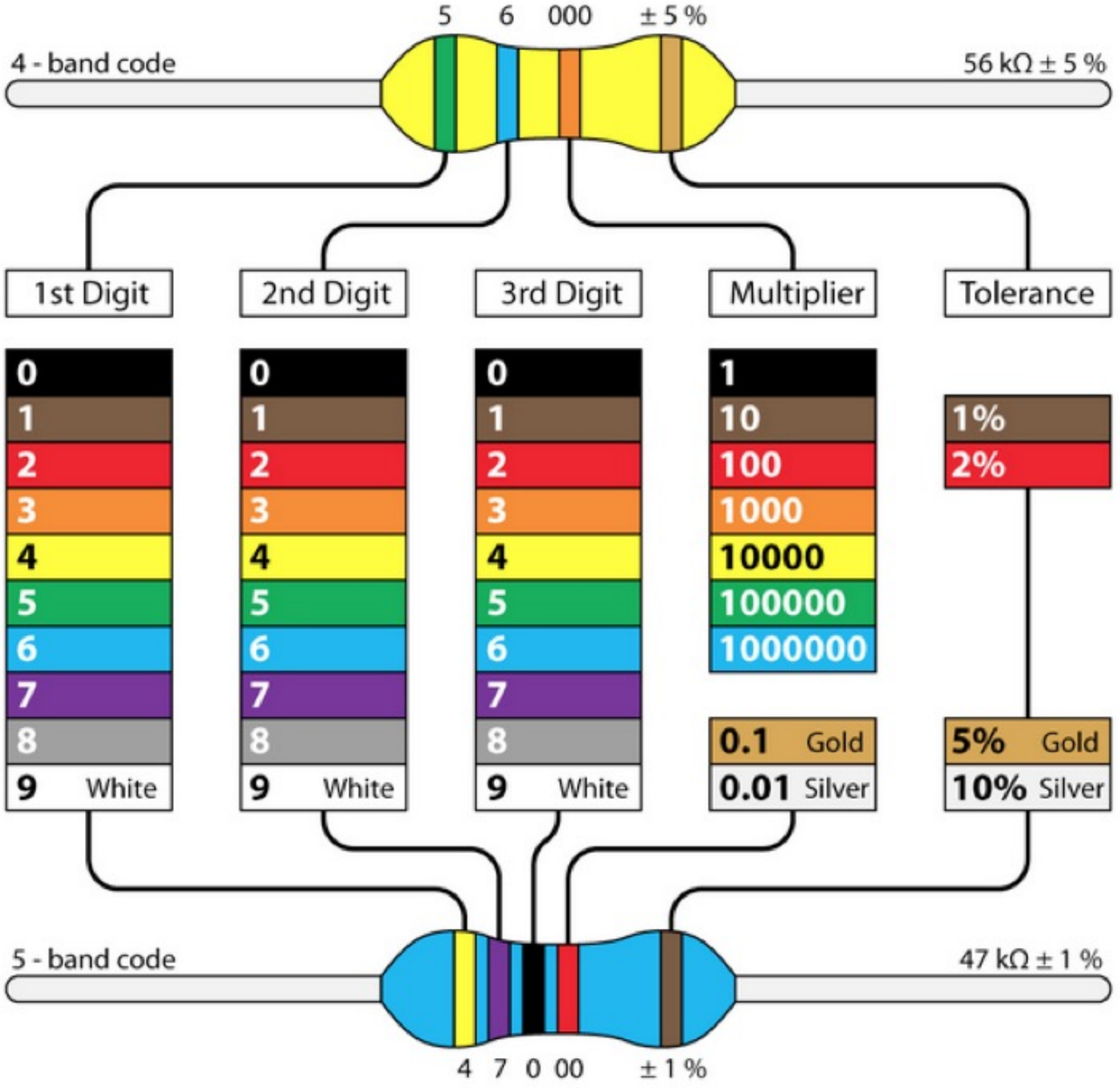
- *Result [0 ... 4095] with 0 → 0V and 4095 → 3.3V*

Photoresistor

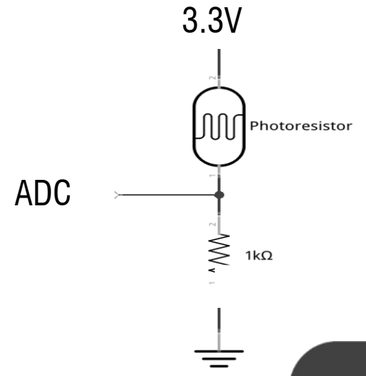




Resistor colour code



analogRead(A0)

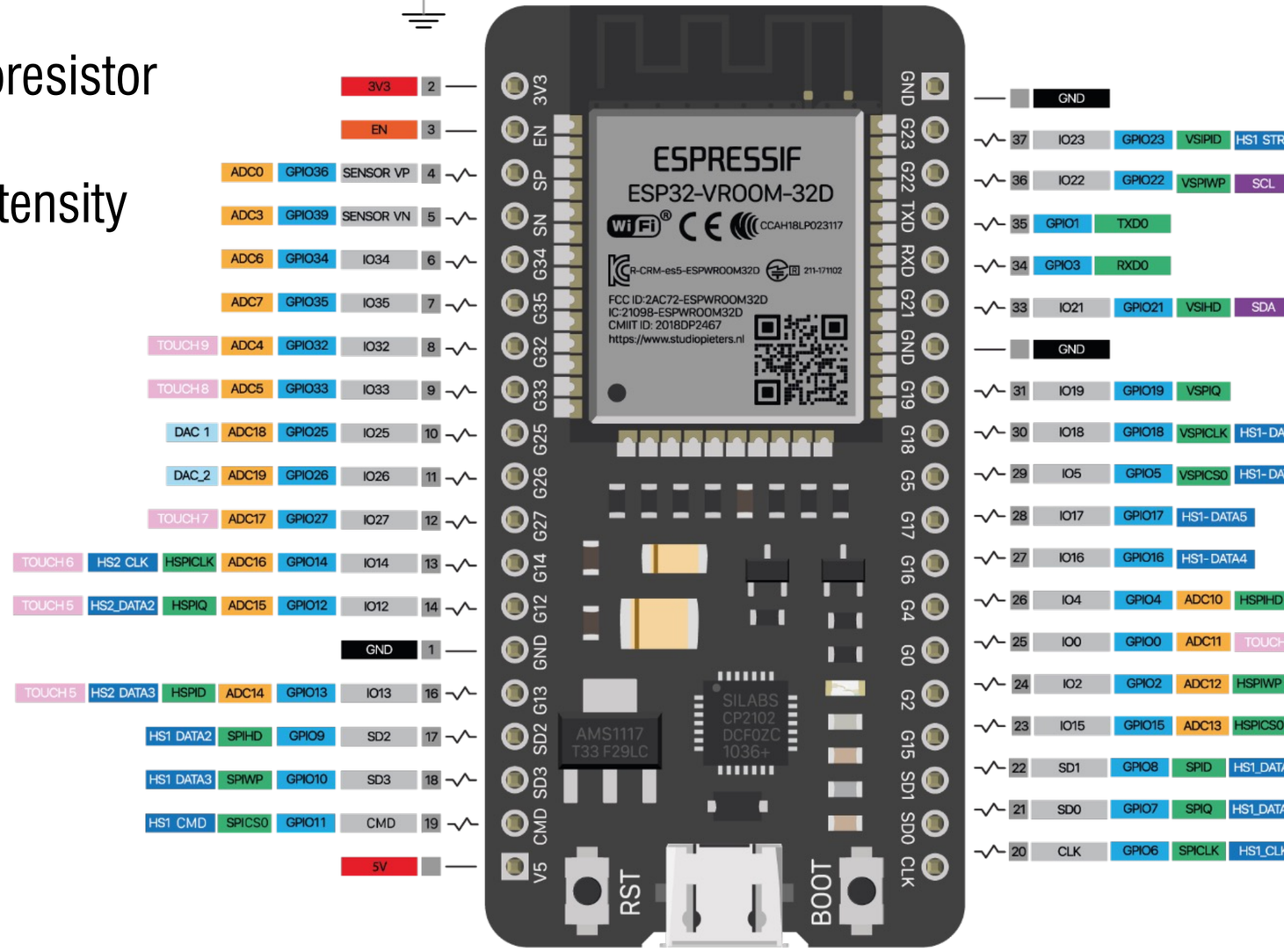


Mini program 2: Print the value of the photoresistor

1. Using photoresistor to sense the light intensity
2. Print out the reading at the same time

Hint:

- Use **GPIO23** as the 3.3V output
- Use **GPIO36** as the ADC pin
- Use the 10K resistor as the voltage divider



Mini program 3: mapping the LED light based on the photoresistor value

1. Read the environmental light with the photoresistor
2. Convert the photoresistor value to the proper range of your LED
3. Map the LED light with the converted value, so that when you cover the photoresistor the LED gets dimmer and vise versa.

Mini program 3: mapping the LED light based on the photoresistor value

1. Read the environmental light with the photoresistor
2. Convert the photoresistor value to the proper range of your LED
3. Map the LED light with the converted value, so that when you cover the photoresistor the LED gets dimmer and vise versa.

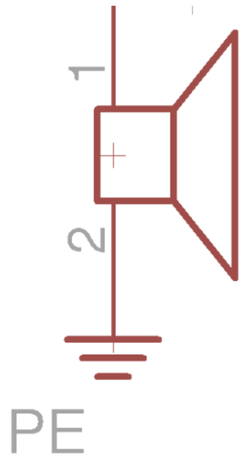
Formatting tools

- int map(value, fromLow, fromHigh, toLow, toHigh)
  - Maps values between [fromLow, fromHigh] and [toLow, toHigh]
  - Lows can be lower than Highs
  - Does not constrain values
- constrain(x, a, b)
  - Constrains **x** to be between **a** and **b**

Sound

Basic setting
- PWM frequency give the tone
- Pulse width give the amplitude

Arduino:
- Start a tone on a pin at a given freq. : tone(pin, frequency)
    - *tone(pin, frequency, duration)*
- Stop a tone on a pin: noTone(pin)



Hint:

- Use GPIO16 for the buzzer
- Check the Arduino examples

Assignment

Light Game

For this assignment:

1. Generate a random number at the beginning of the game
2. The number represents the targeted ambient light intensity
3. Play a simple melody to indicate the beginning of the game
4. The player can now change the ambient light to approximate the target
5. If the number gets closer, the buzzer plays higher pitch
6. If the number gets further from the target, play the lower pitch
7. If the player reaches the target number within 10s, play a simple winning melody
8. If the player loses the game, play a different melody
9. Press a key on your keyboard to restart the game

How to play a simple melody:
https://www.arduino.cc/en/Tutorial/toneMelody


Esc
Tab
Caps Lock