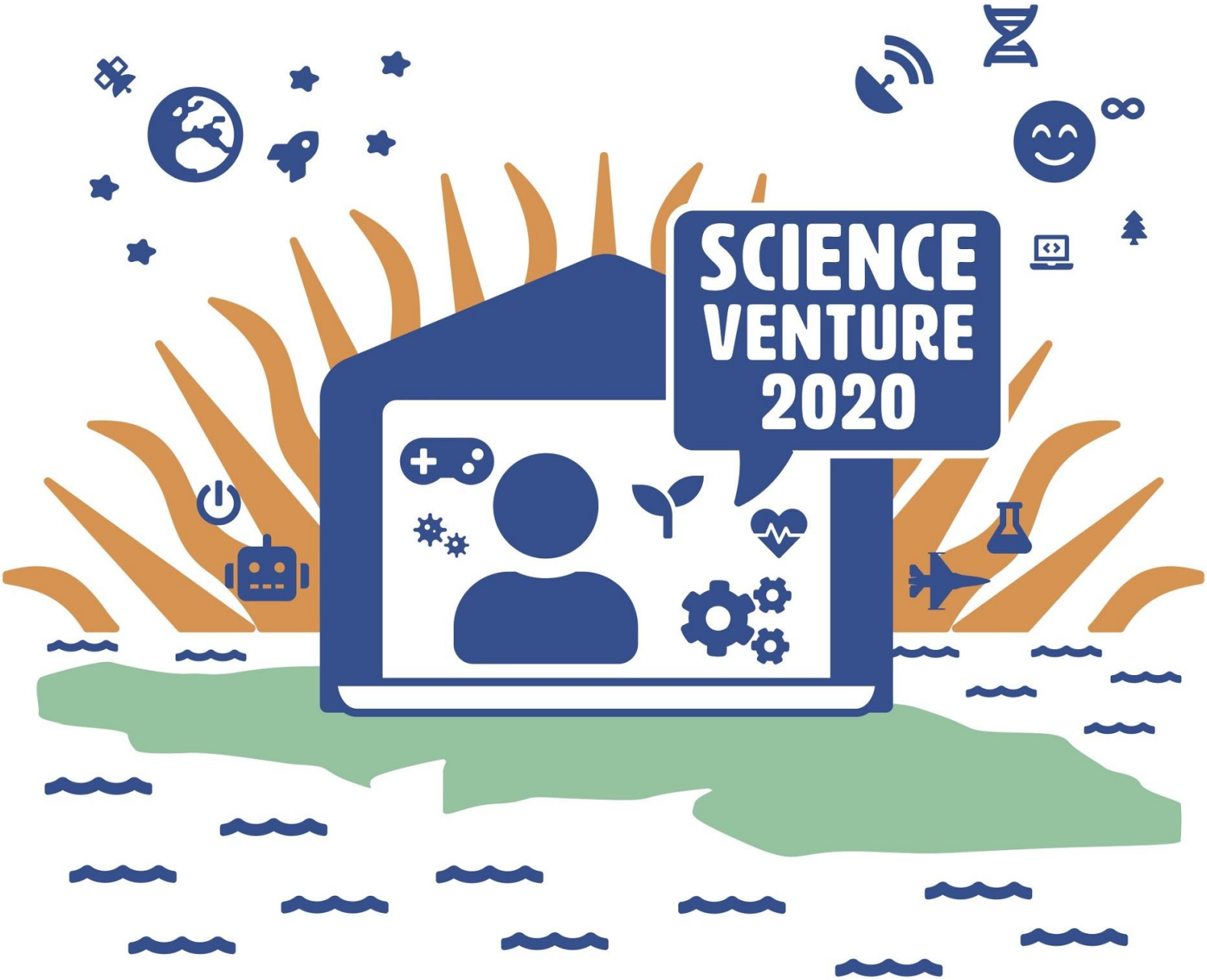


Arduinos

Arduino Reaction Time

Introduce the photoresistor! Read the values of light shining on your Arduino circuit to start a reaction time game. Light an LED for a random amount of time and use button inputs to test your reflexes. Practice wiring the breadboard with multiple circuit components. Create a program to gain points based on your reaction speed, and try to beat your highscore.

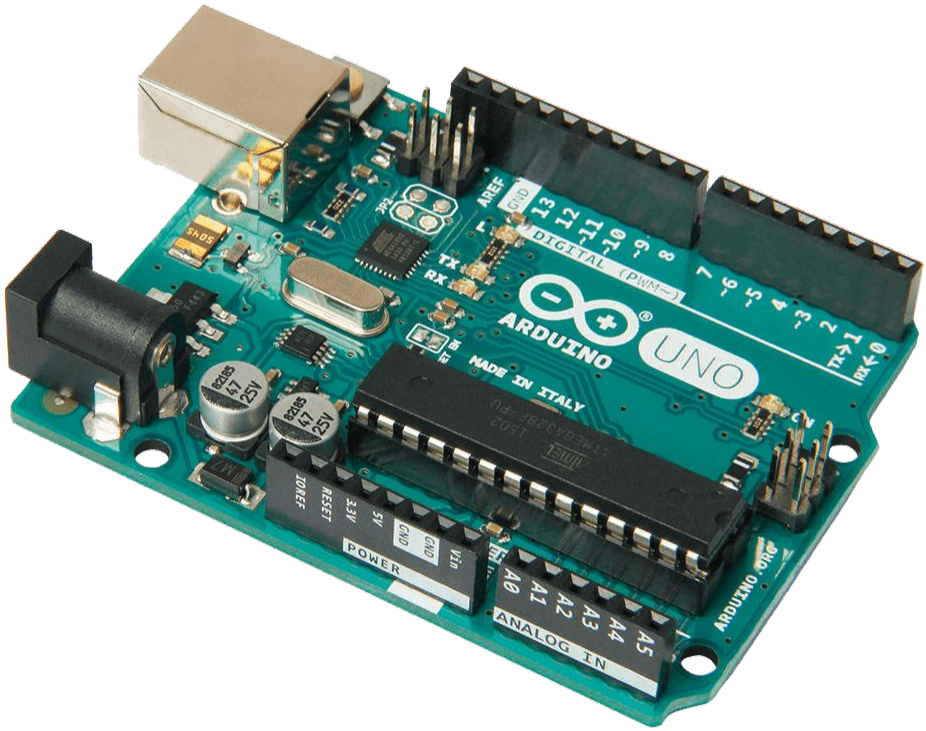
What is a Arduino?

Arduino is an open-source electronics platform based on easy-to-use hardware and software. Arduino boards are able to read inputs - light on a sensor, a finger on a button, or a Twitter message - and turn it into an output - activating a motor, turning on an LED, publishing something online. You can tell your board what to do by sending a set of instructions to the microcontroller on the board. To do so you use the Arduino programming language (based on Wiring), and the Arduino Software (IDE), based on Processing

Arduino coding environment: https://create.arduino.cc/editor/

Materials

- Arduino Uno Kit
  - USB Cable
  - Breadboard
  - LED
  - Button switch
  - Photoresistor
- Computer with Arduino Create



Arduino Reaction Time

Photoresistors - Analog vs. Digital

Photoresistors, also known as LDR (light-dependant resistor), are composed of photo-conductor material which reacts to light. In the absence of light their resistance is very high, when they are exposed to light their resistance is very low. Resistors are used to reduce current flow

The word Photoresistor can be broken down into a combination of two words:

Photo or **Photon** - which means light particles
Resistor - which is used to reduce current

Therefore, the flow of electric current will increase when it is exposed to light and decrease when in the dark.

Because the photoresistor takes a range of values based on light exposure, the readings will be more than just HIGH or LOW. To read these values, we need to use an analog pin instead of a digital pin. Analog pins can intake an electric signal and check the level of voltage received from 0 to 1023 (2^10). This is because of the Arduino's 10 bit Analog to Digital converter (ADC). Digital pins can only interpret a signal as one of two values: HIGH or LOW (0,1)

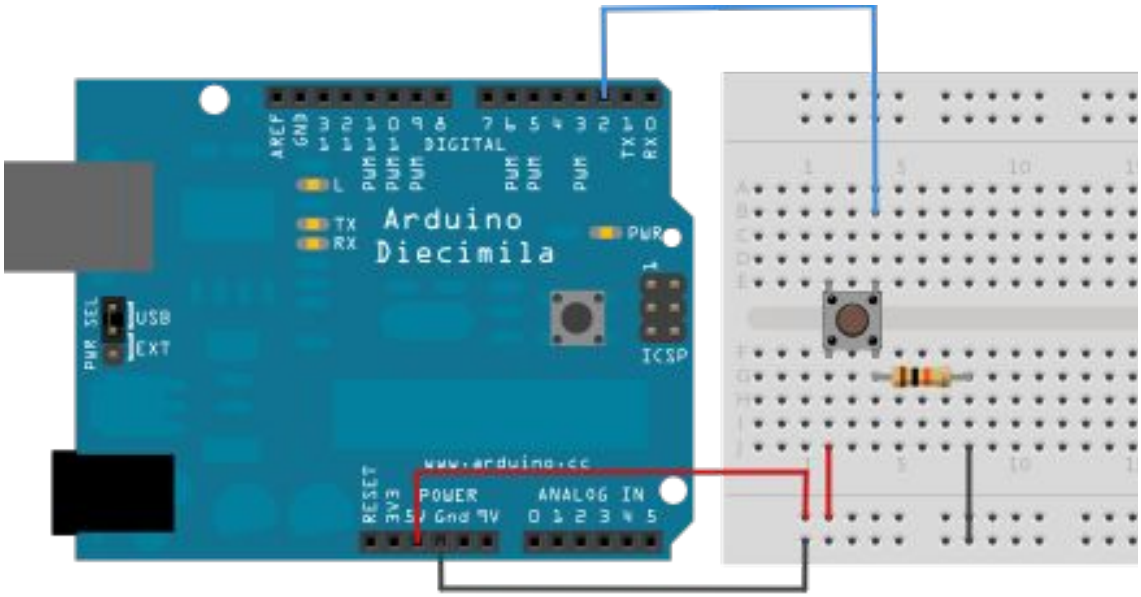
Arduino Reaction Time

Wiring the Circuit

There will be 2 portions of the circuit for this reaction game; one for the button switch and one for the photoresistor. We can use 1 breadboard and put each portion on it's own section of the breadboard. They can share the same power and ground strips on the edge of the breadboard.

Wiring the Button switch (same as the Fibonacci circuit from Day 2)

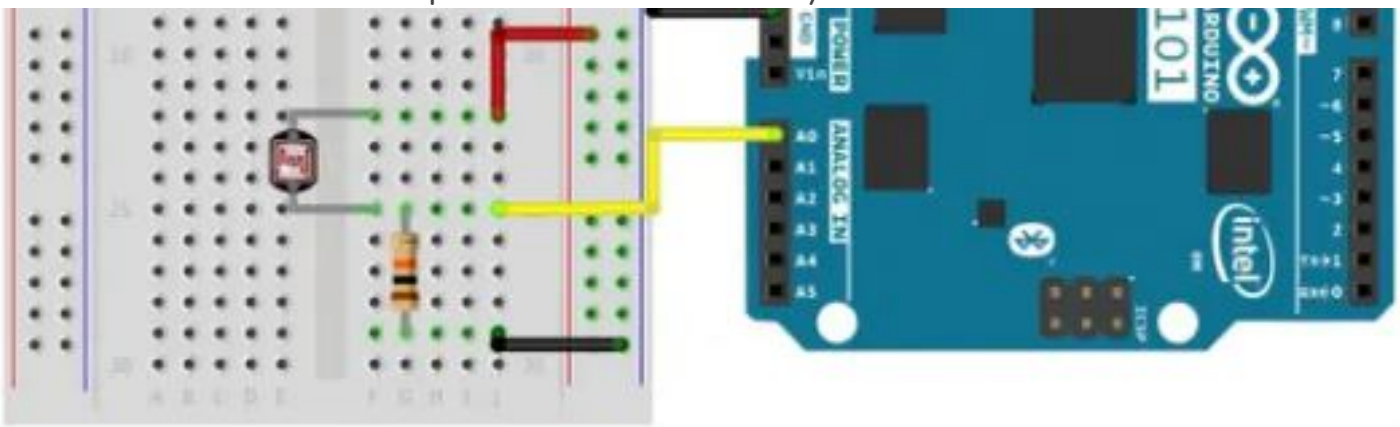
Attach the button across the middle divide of the breadboard. The bottom left corner takes power, the bottom right has a resistor to ground. The top right corner goes to the button pin.



Wiring the Photoresistor

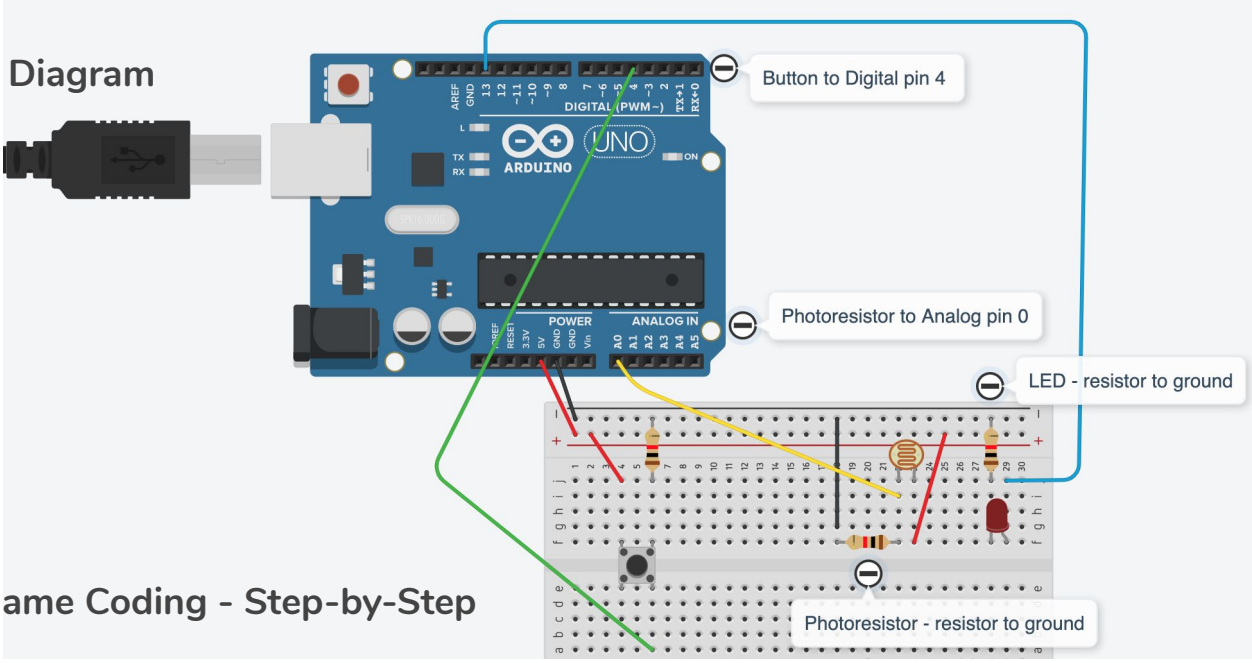
Connect power to one end of the photoresistor. On the other end, we have the lightPin connection and a resistor to ground.

We use an analog pin to read the photoresistor. Connect it to any analog pin, found in the bottom corner of the Arduino pictured here with the yellow connection.



Arduino Reaction Time

Full Circuit Diagram



Reaction Game Coding - Step-by-Step

Initializing Variables

1) Start the code by declaring the variables for the program. Declare constant LED and button pins for use in our program. Declare a buttonState to read our button press.
2) Declare a constant pin for our photoresistor. This must be an analog pin. There are 6 total analog pins (A0 - A5) located on the opposite side of the 13 digital pins.
3) Declare photoresistor variables for Calibration (firstLight), current readings (currentLight) and a threshold value to tell when the photoresistor has gone dark / been covered.

```
const int LED = 13;
const int button = 4;
int buttonState;    // holds the current state of button

const int lightPin = 0; // use an Analog pin for the photoresistor

// photoresistor variables
int firstLight;   // first reading from photoresistor, used to calibrate
int currentLight; // current reading, check to compare with firstLight
int threshold = 50;  // trigger the game once threshold is breached
// adjust the threshold appropriately to photoresistor readings
// if readings are low/unusual, try a different resistor on the photoresistor ground
```

Arduino Reaction Time

5) Declare a randomTime variable for an unpredictable delay. Set a min and max Time, and the random delay will be between these values. A larger range will be more random.

```
//random delay variables
int randomTime;
int minTime = 2000; //minimum random delay (2 seconds)
int maxTime = 5000; //maximum random delay (5 seconds)
```

6) Lastly, declare our score variables. We need a score value when we start and end the game. The difference between these values will give us our points Score in milliseconds, so the lowest score will be the fastest reaction time.
7) We use unsigned long to store the largest number possible (32bits) as we are managing our score by the milliseconds and it can get quite large.

```
//Score variables - use unsigned long for large numbers of milliseconds
unsigned long startScore; //start time when LED turns off
unsigned long endScore; //end time when button is pressed

unsigned long score;  //score = endScore - startScore
```

Void setup() { }

1) Begin the Serial communication. Here we will read our Light values and score.
2) Use analogRead(lightPin); to take the firstLight reading. Keep the photoresistor in a constant lighting environment so we can have a good initial value. Analog pins are used for photoresistors to read values beyond 0 and 1. Print the first value to the serial.

```
void setup() {
  Serial.begin (9600);  //serial monitor begins

//read the inital light value. Keep the photoresistor uncovered
  firstLight = analogRead(lightPin);
  Serial.println(firstLight);
```

Arduino Reaction Time

3) Set the pin modes. We require an input to read the button press, and output for LED.
4) Randomize the randomTime delay variable by using random(minTime, maxTime); This randomizes the value anywhere between min and max.
5) Delay to finish setup. By now you have code to see your first light reading, so it is a good idea to test and see if your value is in a good range (50-300). If the value is too low or unusual, try using a different resistor when grounding the photoresistor.

```
 pinMode(button, INPUT); //set button input
 pinMode(LED, OUTPUT);   //set LED output

//randomize the delay time between min and max
 randomTime = random(minTime, maxTime);
 delay(2000);              // give the serial connection time to start
}
```

Void loop() { }

1) Start by reading the current light value and printing to serial. This will let us check the values before we cover the photoresistor to start the game.

```
void loop() {
 //read current light values and print to Serial
 currentLight = analogRead(lightPin);
 Serial.println(currentLight);
```

2) if the currentLight value reads below the threshold difference from the initial value set in setup, then the game will begin. Here may be a good idea to change the threshold based on what readings you are getting. In heavily lit environments, threshold may be around 100 but in dim lighting it may be 25. Find what works for your space.
3) Once game has started, write LED to HIGH. This shows us the game has begun.
4) Delay by randomTime. We don't know when the reaction time will start.
5) Read the button - this checks if we are already pressing the button before the time starts
6) Use millis(); to read our start time into startScore. This function reads the milliseconds since the program has started.

Arduino Reaction Time

7) Using an if statement, check if the button has been pressed before the LED is LOW. We want to stop players from holding the button until the game starts. Include a message to indicate we had a false start. Delay and set startScore = 0;. Close the if statement.
8) Turn off the LED. This is our signal that the game has started.

```
  // Turn on LED when we cover the photoresistor
 if (currentLight < firstLight - threshold) {

   digitalWrite(LED, HIGH);

// wait a random amount of time before LED turns off and game starts
   delay(randomTime);

   buttonState = digitalRead(button);   // read the button state
   startScore = millis();             // read the starting time

// Check if button is pressed before the game has started
   if (buttonState == HIGH) {
     Serial.println("Wait until the LED is off to press the button!");
     delay(1000);
     Serial.println("No points for you!");
     startScore = 0; // reset initial points
   }
   digitalWrite(LED, LOW); //LED off, game has begun
```

Arduino Reaction Time

9) Now the game has started. Use a while loop to keep our program in the game while we wait for the button to be pressed. Read the button pin so we know once it is pressed.

10) If buttonState changes, set the endScore millis(); and perform the score calculation. Subtracting startScore from the endScore will give us the milliseconds between when the LED turned off and when we pushed the button. This is our score.

```
// My button goes HIGH when pressed. If your button is not working,
// try swapping the LOW and HIGH from the while and if loop arguments
    while (buttonState == LOW) {          // while button NOT pressed
      buttonState = digitalRead(button);    // read the button state
      if (buttonState == HIGH) {          // when button pressed
        endScore = millis();              // read end time
```

11) Print the score and delay a few seconds so we have time to read the serial monitor.

```
// subtract start time from end to find the difference. This is your score.
        score = endScore - startScore;
        Serial.print("Your score is: ");
        Serial.println(score);   //prints score
        delay(2000);
      } // close button if statement
    } // close button while loop
  } // close photoresistor if statement

// reset random delay to a new random value
  randomTime = random(minTime, maxTime);
  delay(1000);
} // close void loop
```

12) Close the button if statement, while loop, and photoresistor if statement. Reassign the randomTime variable to a new random delay before closing the void loop.

13) This finishes our program. Test your reaction time and try to get the lowest score.

Arduino Reaction Time

reactionGame.ino - Complete Code

```
/*
Cover the Photoresistor to start the game and turn on the LED.
The LED will stay on for a random amount of time.
Press the button once the LED has turned off to test your reaction time.
Read your score in the serial monitor, this is your reaction time in milliseconds!
*/

// declare the Pin constants for the button, LED and photoresistor pins
const int button = 4;
const int LED = 13;
const int lightPin = 0; // use an Analog pin for the photoresistor

//photoresistor variables
int firstLight;   // first reading from photoresistor, used to calibrate
int currentLight; // current reading, check to compare with firstLight

int threshold = 50;  // trigger the game once threshold is breached
//adjust the threshold appropriately to photoresistor readings
//if readings are low/unusual, try a different resistor on the photoresistor ground

//random delay variables
int randomTime;
int minTime = 2000; //minimum random delay (2 seconds)
int maxTime = 5000; //maximum random delay (5 seconds)

//Score variables - use unsigned long for large numbers of milliseconds
unsigned long startScore; //start time when LED turns off
unsigned long endScore; //end time when button is pressed

unsigned long score;  //score = endScore - startScore

int buttonState;    // holds the current state of button
```

Arduino Reaction Time

```
 void setup() {
  Serial.begin (9600);  //serial monitor begins

//read the inital light value. Keep the photoresistor uncovered
  firstLight = analogRead(lightPin);
  Serial.println(firstLight);

  pinMode(button, INPUT); //set button input
  pinMode(LED, OUTPUT);   //set LED output

//randomize the delay time between min and max
  randomTime = random(minTime, maxTime);
  delay(2000);              // give the serial connection time to start
}

void loop() {
  //read current light values and print to Serial
  currentLight = analogRead(lightPin);
  Serial.println(currentLight);

  // Turn on LED when we cover the photoresistor
  if (currentLight < firstLight - threshold) {

    digitalWrite(LED, HIGH);

// wait a random amount of time before LED turns off and game starts
    delay(randomTime);

    buttonState = digitalRead(button);   // read the button state
    startScore = millis();            // read the starting time
```

Arduino Reaction Time

```
// Check if button is pressed before the game has started
    if (buttonState == HIGH) {
      Serial.println("Wait until the LED is off to press the button!");
      delay(1000);
      Serial.println("No points for you!");
      startScore = 0; // reset initial points
    }
    digitalWrite(LED, LOW); //LED off, game has begun

// My button goes HIGH when pressed. If your button is not working,
// try swapping the LOW and HIGH from the while and if loop arguments
    while (buttonState == LOW) {          // while button NOT pressed
      buttonState = digitalRead(button);    // read the button state
      if (buttonState == HIGH) {          // when button pressed
        endScore = millis();              // read end time

// subtract start time from end to find the difference. This is your score.
        score = endScore - startScore;
        Serial.print("Your score is: ");
        Serial.println(score);   //prints score
        delay(2000);

      } // close button pressed
    } // close button unpressed
  } // close photoresistor if statement

// reset random delay to a new random value
  randomTime = random(minTime, maxTime);
  delay(1000);
} // close void loop
```

Arduino Reaction Time

Extensions

Add a highscore to your reaction game.

1) Declare a highScore variable in the intialization section at the start of the program. Set a large highScore so we can easily beat it with our reaction time.

```
unsigned long highScore = 100000; //set a large highscore
```

2) After calculating and printing your score in the void loop, check if there is a new highscore. The highscore is the lowest number of milliseconds, so we can check if the score is less than the previous highscore. If yes, set the new highscore to our current score. Print a victory message to the Serial.
3) I have included a for loop to flash the LED. This is just a flashy celebration loop, you can replace this loop with any celebration code you like.
4) else { Print a message to let us know the highscore was not beaten.
5) Outside of the if loop, delay and print the current highscore.
6) That is all we need to add for our High score extension! Continue the program as before, by closing the game loops and resetting the random delay.

```
if (score < highScore) {  //if new score is lower than previous highscore
      highScore = score;      //set new lowest score
      Serial.println("New high score!");
      for (int i = 0; i < 9; i++) { // celebrate a highscore by
        digitalWrite(LED, HIGH);    // flashing the LED
        delay(100);
        digitalWrite(LED, LOW);
        delay(100);
      }
    } else {                    //if not faster than highscore
      Serial.println("Gotta be faster than that!");
    }

    delay(2000);  // wait before printing the highscore to the Serial monitor
    Serial.print("The current high score is: ");
    Serial.println(highScore);  // print fastest reaction time
```

Arduino Reaction Time

reactionGameHighScore.ino - Complete Code with High Score extension

```
/*
Cover the Photoresistor to start the game and turn on the LED.
The LED will stay on for a random amount of time.
Press the button once the LED has turned off to test your reaction time.
Read your score in the serial monitor, this is your reaction time in milliseconds!
*/

// declare the Pin constants for the button, LED and photoresistor pins
const int button = 4;
const int LED = 13;
const int lightPin = 0; // use an Analog pin for the photoresistor

//photoresistor variables
int firstLight;   // first reading from photoresistor, used to calibrate
int currentLight; // current reading, check to compare with firstLight
int threshold = 50;  // trigger the game once threshold is breached
//adjust the threshold appropriately to photoresistor readings
//if readings are low/unusual, try a different resistor on the photoresistor ground

//random delay variables
int randomTime;
int minTime = 2000; //minimum random delay (2 seconds)
int maxTime = 5000; //maximum random delay (5 seconds)

//Score variables - use unsigned long for large numbers of milliseconds
unsigned long startScore; //start time when LED turns off
unsigned long endScore; //end time when button is pressed

unsigned long score;  //score = endScore - startScore
unsigned long highScore = 100000; //set a large highscore

int buttonState;    // holds the current state of button
```

Arduino Reaction Time

```
void setup() {
  Serial.begin (9600);  //serial monitor begins

//read the inital light value. Keep the photoresistor uncovered
  firstLight = analogRead(lightPin);
  Serial.println(firstLight);

  pinMode(button, INPUT); //set button input
  pinMode(LED, OUTPUT);   //set LED output

//randomize the delay time between min and max
  randomTime = random(minTime, maxTime);
  delay(2000);              // give the serial connection time to start
}

void loop() {
  //read current light values and print to Serial
  currentLight = analogRead(lightPin);
  Serial.println(currentLight);

  // Turn on LED when we cover the photoresistor
  if (currentLight < firstLight - threshold) {

    digitalWrite(LED, HIGH);

// wait a random amount of time before LED turns off and game starts
    delay(randomTime);

    buttonState = digitalRead(button);   // read the button state
    startScore = millis();            // read the starting time

// Check if button is pressed before the game has started
    if (buttonState == HIGH) {
      Serial.println("Wait until the LED is off to press the button!");
      delay(1000);
      Serial.println("No points for you!");
      startScore = 0; // reset initial points
    }
    digitalWrite(LED, LOW); //LED off, game has begun
```

Arduino Reaction Time

```
// My button goes HIGH when pressed. If your button is not working,
// try swapping the LOW and HIGH from the while and if loop arguments
    while (buttonState == LOW) {          // while button NOT pressed
      buttonState = digitalRead(button);    // read the button state
      if (buttonState == HIGH) {          // when button pressed
        endScore = millis();              // read end time

// subtract start time from end to find the difference. This is your score.
        score = endScore - startScore;
        Serial.print("Your score is: ");
        Serial.println(score);   //prints score
        delay(2000);

        if (score < highScore) {  //if new score is lower than previous highscore
          highScore = score;      //set new lowest score
          Serial.println("New high score!");
          for (int i = 0; i < 9; i++) { // celebrate a highscore by
            digitalWrite(LED, HIGH);    // flashing the LED
            delay(100);
            digitalWrite(LED, LOW);
            delay(100);
          }
        } else {                    //if not faster than highscore
          Serial.println("Gotta be faster than that!");
        }

        delay(2000);  // wait before printing the highscore to the Serial monitor
        Serial.print("The current high score is: ");
        Serial.println(highScore);  // print fastest reaction time
        delay(1000);
      }
    }
  }

// reset random delay to a new random value
  randomTime = random(minTime, maxTime);
  delay(1000);
}
```

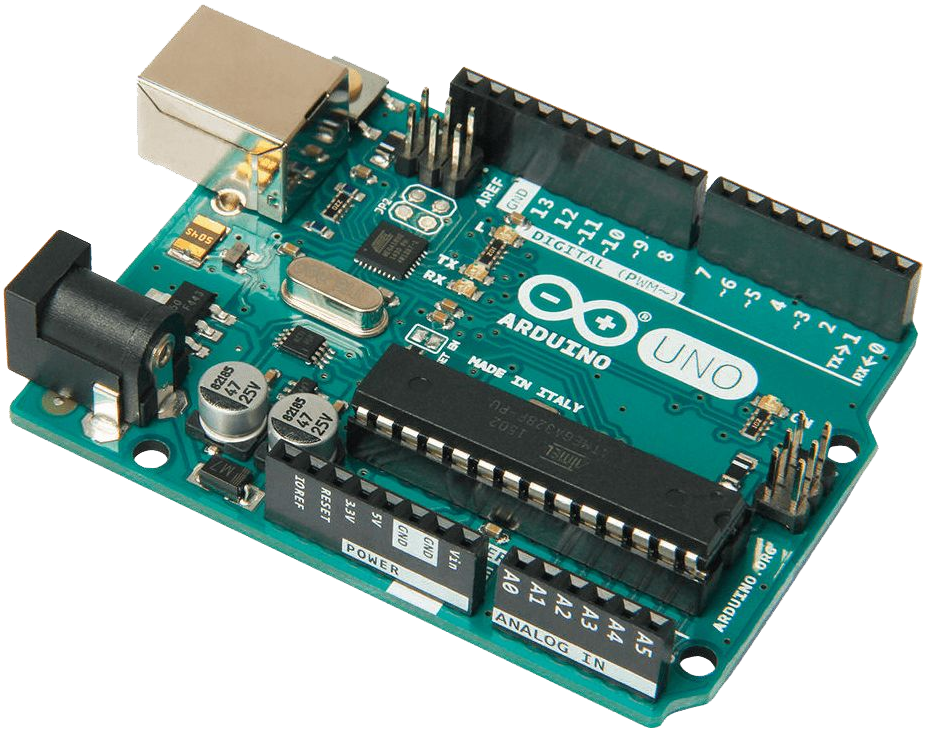
Arduino Reaction Time

Extensions

Also try,
Widening the random interval of delay to make the game less predictable

Have a fixed number of rounds and score points depending on your average reaction time across all rounds.

Add a check to make sure you are beating your highscore, or else the game will end. See how many rounds you can play while improving your reaction time each round.



#SVatHome

Want to share your project or results with us?

Email or tag us @ScienceVenture

Have a question?

Reach us at svcamp@engr.uvic.ca