Arduino Reaction Game Project	Tina Götschi	tina@adacollege.org.uk
https://github.com/MsGotschi/arduinoReactionGame		

This is based on the classic WHACK A MOLE game and shows students the types of electronics and programming that goes into a funfair game that they might be familiar with

It can be used to teach the basics of the Arduino and programming and is endlessly extensible. Here's a suggested outline:

Part 1: Blink an LED

Part 2: Control the LED with a button

Part 2a: Use a BIG DOME PUSH button

Part 3: Debug messages to the Serial Monitor

Part 4: De-bounce the button Press

Part 5: Create a random delay

Part 6: Put the game together - timing

Part 7: Add a Liquid Crystal Display (LCD) for messages

You will need:

The Arduino Starter kit

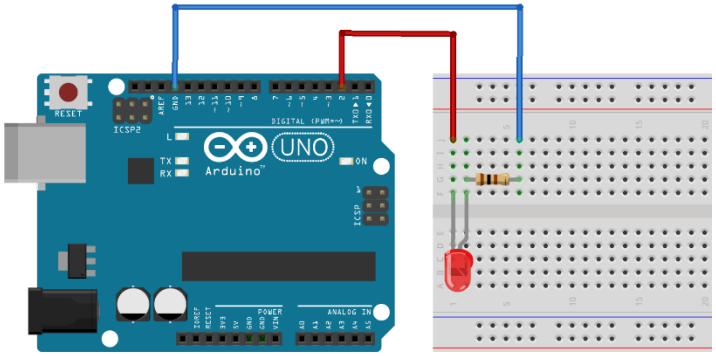
EXTRA: Big Dome Push Button and a box

https://www.proto-pic.co.uk/big-dome-pus

Part 1: Blink an LED

Use the built in Example sketch

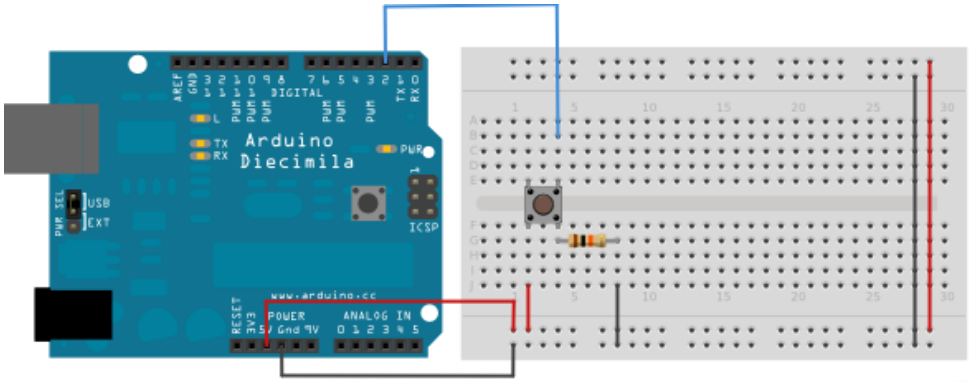
https://www.arduino.cc/en/Tutorial/Blink



```
// the setup function runs once when you press reset or power the board
void setup() {
 // initialize digital pin 13 as an output.
 pinMode(13, OUTPUT);
}

// the loop function runs over and over again forever
void loop() {
 digitalWrite(13, HIGH);

// turn the LED on (HIGH is the voltage level)
 delay(1000);

 // wait for a second
 digitalWrite(13, LOW);

// turn the LED off by making the voltage LOW
 delay(1000);

 // wait for a second
}
```

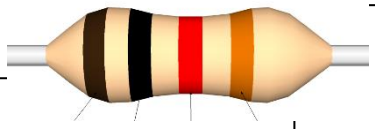
Part 2: Control the LED with a button

https://www.arduino.cc/en/Tutorial/Button



NOTE:

The resistor acts as a PULL DOWN resistor to ensure when the button is not pressed the signal is consistently 0V

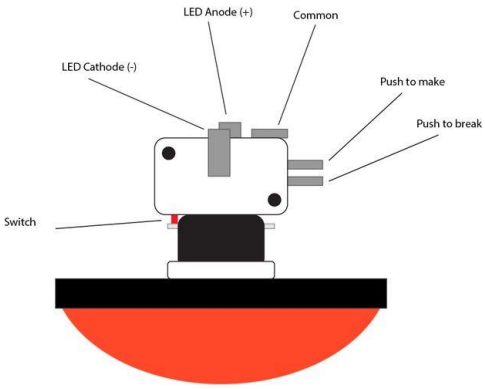


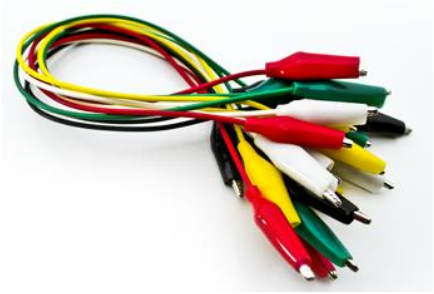
```
// constants won't change. They're used here to
// set pin numbers:
const int buttonPin = 2;    // the number of the pushbutton pin
const int ledPin =  13;     // the number of the LED pin

// variables will change:
int buttonState = 0;        // variable for reading the pushbutton status

void setup() {
 // initialize the LED pin as an output:
 pinMode(ledPin, OUTPUT);
 // initialize the pushbutton pin as an input:
 pinMode(buttonPin, INPUT);
}

void loop() {
 // read the state of the pushbutton value:
 buttonState = digitalRead(buttonPin);

 // check if the pushbutton is pressed.
 // if it is, the buttonState is HIGH:
 if (buttonState == HIGH) {
   // turn LED on:
   digitalWrite(ledPin, HIGH);
 } else {
   // turn LED off:
   digitalWrite(ledPin, LOW);
 }
}
```

EXTENSION: Use the LED in Part 1 instead of the built in LED on pin 13

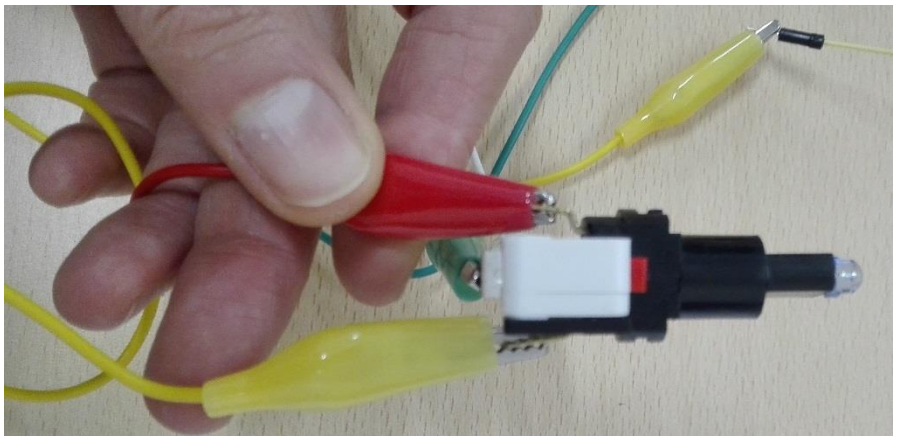
Part 2a: Use a BIG DOME PUSH button

The big dome push button includes an LED and a push button.





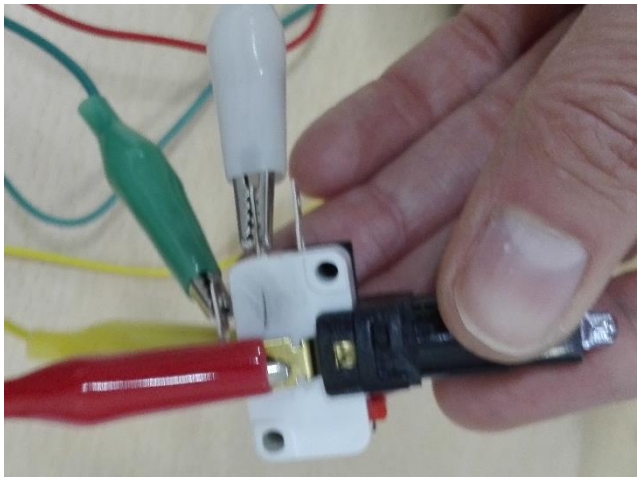
HINT: Crocodile clips are useful for connecting this. Alternatively you will need to crimp some spade connectors (female) onto wires



Connecting the LED

Positive to PIN and Ground

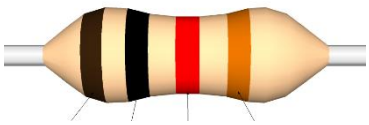
There is a built-in resistor so it won't blow!



Connecting the BUTTON

Bottom connector (COMMON) to power

Push to break connector to PIN but also connected via pull down resistor (10Ohm) to GROUND

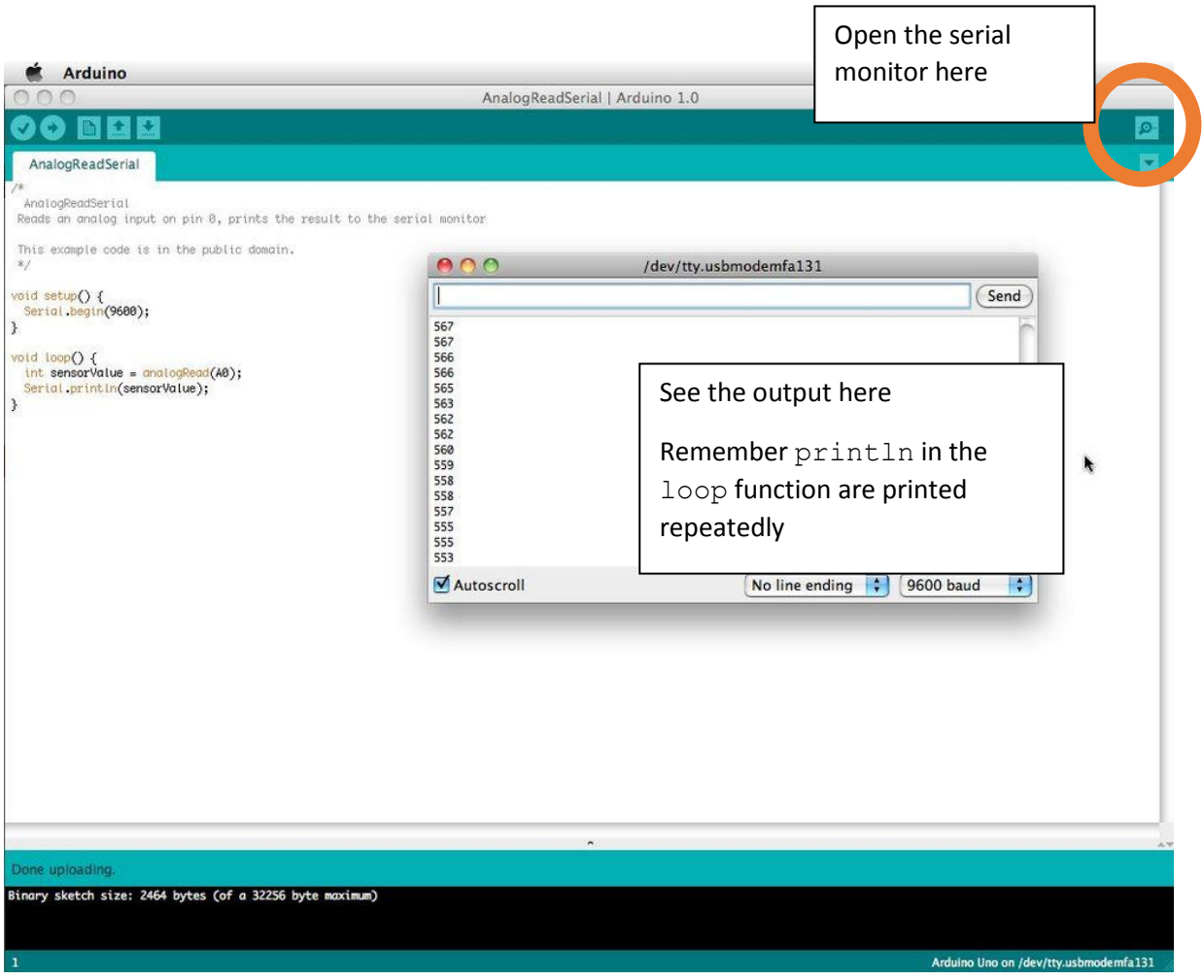


Part 3: Debug messages to the Serial Monitor

You might want to see what is going on in your code. The Arduino's Serial Monitor is useful for this purpose.

```
void setup() {
 Serial.begin(9600);     // open the serial port at 9600 bps:
}

void loop() {
 // print labels
 Serial.print("hello world... ");      // prints a string. Stays on this line
 Serial.println(" now go to a new line...");

}
```

https://www.arduino.cc/en/Serial/Print



Part 4: De-bounce the button Press

Pushbuttons often generate spurious open/close transitions when pressed, due to mechanical and physical issues: these transitions may be read as multiple presses in a very short time fooling the program.

https://www.arduino.cc/en/Tutorial/Debounce

Part 5: Create a random delay

https://www.arduino.cc/en/Reference/Random

```
long randNumber;

void setup(){
  Serial.begin(9600);
  // if analog input pin 0 is unconnected, random analog
  // noise will cause the call to randomSeed() to generate
  // different seed numbers each time the sketch runs.
  // randomSeed() will then shuffle the random function.
  randomSeed(analogRead(0));
}

void loop() {
  // print a random number from 0 to 299
  randNumber = random(300);
  Serial.println(randNumber);

  // print a random number from 10 to 19
  randNumber = random(10, 20);
  Serial.println(randNumber);

  delay(randNumber*1000); // wait for that random amount of time
}
```

Part 6: Put the game together - timing

This game will time how long it took you to “whack” the “mole” – ie hit the button.

The Arduino board, like all computers, measures time as a number of milliseconds since a certain time. For the Arduino it since the board began running the current program. This number will overflow (go back to zero), after approximately 50 days.

https://www.arduino.cc/en/Reference/Millis

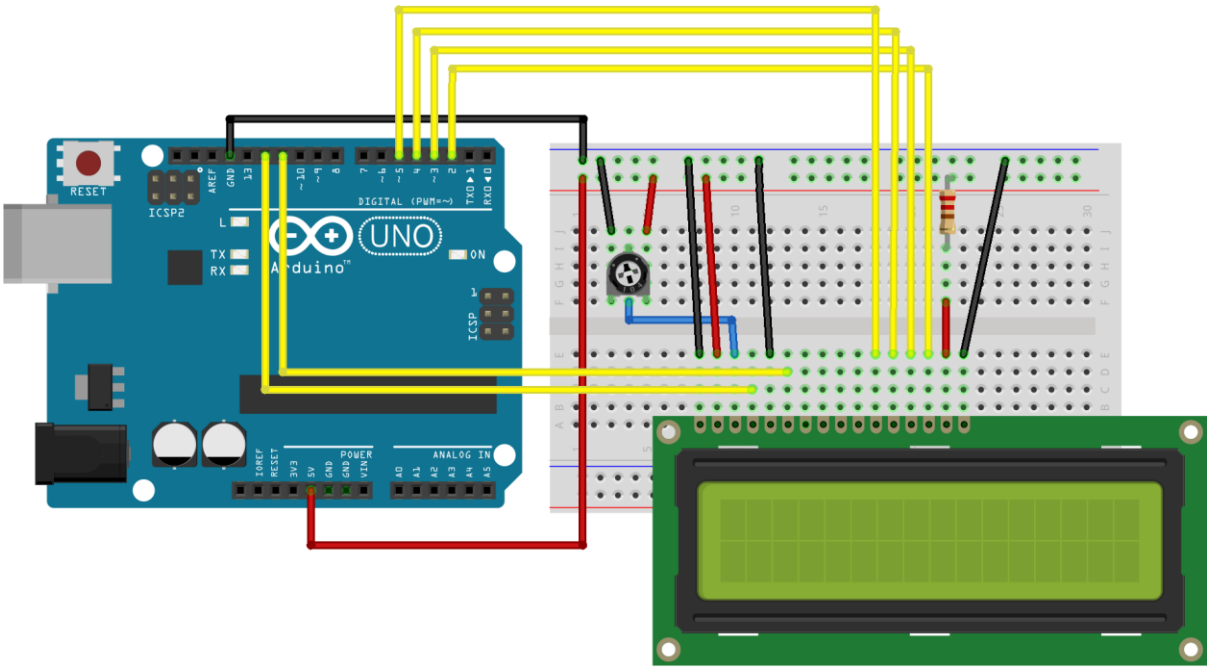
So, to time an event, you need to take a time check before the event begins (when the light goes on) and then when the event ends (the button is pressed and the light is switched off) and find the difference between the 2. Remember, this will be a value in milliseconds so divide by 1000 to get seconds.

```
unsigned long startTime;

unsigned long endTime;

void setup(){
 Serial.begin(9600);
}
void loop(){


 randNumber= random(5000);

 startTime = millis();

  delay(randNumber);   //wait for a random time
 endTime = millis();

 Serial.println("Random Time wait was= ", endTime - startTime);

 // wait a second so as not to send massive amounts of data

delay(1000);

}
```

Part 7: Add a Liquid Crystal Display (LCD) for messages

LCD displays are a nice way to finish off you game. Once you have the big dome push button and the display you can hide the Arduino and breadboard inside the box and present the user with a easy to use interface a bit like the one below. Plus you can get really creative with the box – decorate it yourself!

https://www.arduino.cc/en/Reference/LiquidCrystal



Liquid Crystal Tutorials

Hello World! – How to wire an LCD display and bring it to life.
Blink - Control of the block-style cursor.
Cursor - Control of the underscore-style cursor.
Display - Quickly blank the display without losing what's on it.
TextDirection - Control which way text flows from the cursor.
Scroll - Scroll text left and right.
Serial display - Accepts serial input, displays it.
SetCursor - Set the cursor position.

Final Completed Code (one example – may have bugs and can probably be improved!)

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(12,11,5,4,3,2);

int LED = 8;
int BUTTON = 7;
int timelapse = 0;
boolean lastButton = LOW;
boolean currentButton = LOW;
boolean message;
boolean started = false;
boolean timer = false;
long startTime;
long endTime;
long randomTime;
float elapsedTime;

void setup() {
  // put your setup code here, to run once:
pinMode(LED,OUTPUT);
digitalWrite(LED, LOW);

pinMode(BUTTON,INPUT);

lcd.begin(16,2);
Serial.begin(9600);


lcd.print("Start?");
lcd.setCursor(0,1);
lcd.print("Press button");

int go = 0;
while (go == 0) {
  go =digitalRead(BUTTON);
}
lcd.setCursor(0,0);
lcd.print("REACTION GAME GO!");
lcd.setCursor(0,1);
lcd.print("                ");
delay(1000);
lcd.setCursor(0,0);
lcd.print("Press the button ");
delay(1000);
lcd.setCursor(0,0);
lcd.print("when the light   ");
delay(1000);
lcd.setCursor(0,0);
lcd.print("turns ON........");
delay(2000);

}
```

```
void loop() {
  // put your main code here, to run repeatedly:

  lcd.noDisplay(); // wipe out instructions


  //wait for a button press

  while (digitalRead(BUTTON)== LOW) {

  }

  lcd.display();

  started = !started; // we've started the game!! or ended the game!!


  if (started == true && timer == false) {
          //start the random timer
          randomTime = random(4,10)*1000; //get a random number btw 4
and 10, *1000 to seconds


          //switch the light on once to say the game is starting
          message = false;
          if (!message) {
            lcd.setCursor(0,0);
            lcd.print("game has started   ");
            lcd.setCursor(0,1);
            lcd.print("                   ");
            delay(500);
            message = true;
          }

          digitalWrite(LED, HIGH);
          delay(100);
          digitalWrite(LED,LOW);

          //wait for a random amount of time
          delay(randomTime);

          //now start timing the player
          startTime=millis();
          //switch the light on

          digitalWrite(LED,HIGH);
          //start timing
           timer = true;

    }//game started
```

```
   if (started == true && timer == true) {
    //the game has started and we are timing, so we must be waiting for
the button press!

    while (digitalRead(BUTTON)== LOW) {

    }
    // button was pressed... let's stop and work out elaspsed time
    started=false;

   }

   if (started == false && timer == true) {
        digitalWrite(LED,LOW);
        // this means they are being timed, and the button press has
arrived since !
        endTime=millis();
        elapsedTime= endTime-startTime;
        elapsedTime= elapsedTime/1000;
        lcd.setCursor(0,0);
        lcd.print("you took         ");
        lcd.setCursor(0,1);
        lcd.print(elapsedTime);
        delay(2000); // wait a bit before the game starts again...

        timer=false; // go back to initial state
    }//timing



}//  main loop
```

EXTENTIONS:

Make a Whack a mole game with 16 push buttons and an Arduino Mega (and soldering)
https://www.youtube.com/watch?v=T1jdjmbe1mM&feature=youtu.be&t=2m38s

https://www.arduino.cc/en/Main/ArduinoBoardMega2560

Another tutorial:

https://create.arduino.cc/projecthub/Blue_jack/measure-your-reaction-time-d68f90