AuE 835
Automotive Electronics Integration

PROJECT 1: EMBEDDED SYSTEM AND AUTONOMOUS VEHICLES

Project Schedule

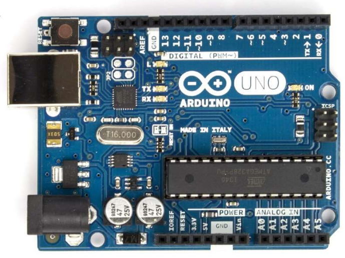
- Oct 18 – Arduino and programming
- Oct 23 – Ultrasonic sensing, vehicle control & Project 1 Announcement
- Oct 25 – Signal processing review and Project 1 hands-on
- Oct 30 – Control review and Project 1 hands-on
- Nov 1 - Project 1 debugging, Q&A, Test details
- Nov 8 - Project 1 Test

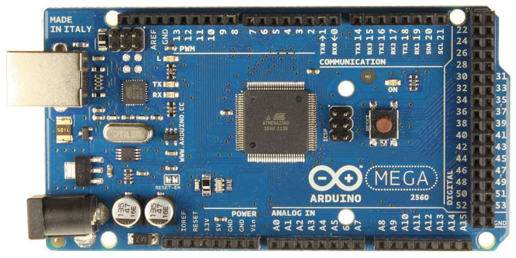
- Nov 13 – Autonomous boat control & Project 2 Announcement
- Nov 15 – Project 2 debugging, Q&A, Test details
- Nov 20 – Project 2 debugging, Q&A, Test details
- Nov 27 – Project 2 Test

- Presentations and report writing

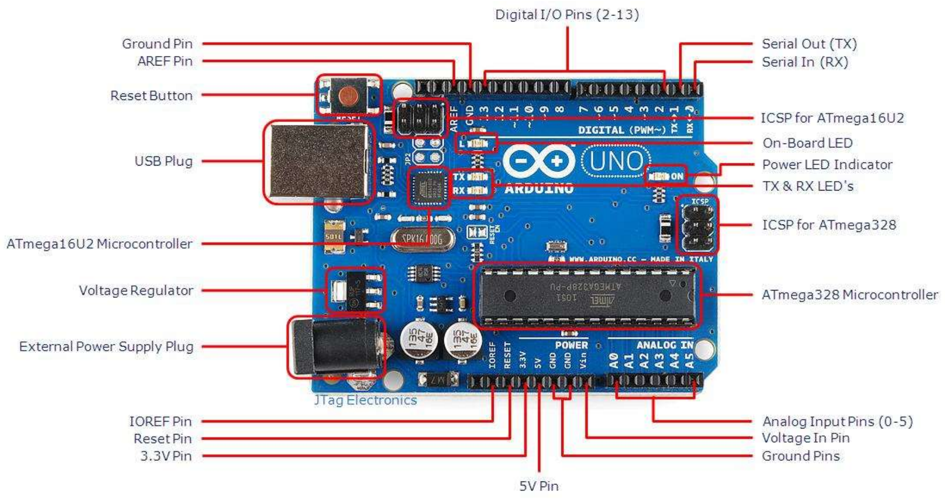
Embedded System and Arduino

- An embedded system is a computer system with a dedicated function within a larger mechanical or electrical system, often with real-time computing constraints.
- Arduino is an open-source embedded system based on easy-to-use hardware and software (https://www.Arduino.cc).

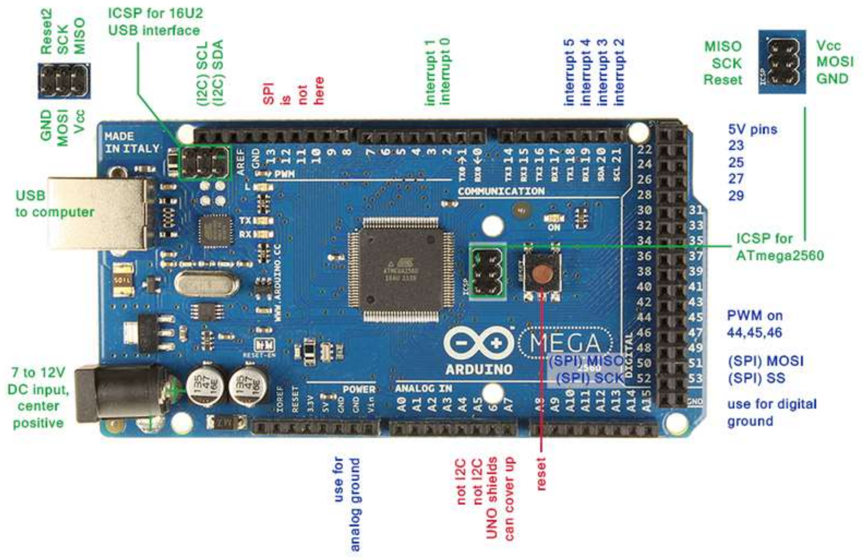

Arduino Uno R3


Arduino Mega 2560

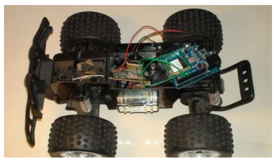
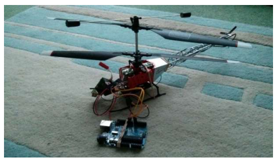
Arduino Uno R3

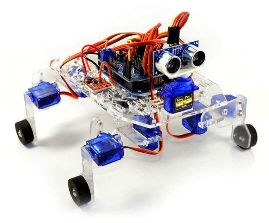
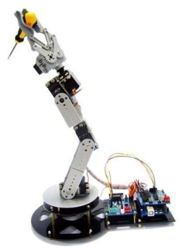


Arduino Mega 2560



Fun Projects with Arduino

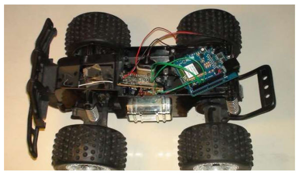
Fun Projects with Arduino

Forte mobile robotic platform (controlled by Arduino Mega 2560)

7

Why Arduino for AuE835?

❖ Arduino is a fast prototyping tool for studying electronics, signal processing and controls.



❖ Limitation?

❖ It is useless without anything attached to it (sensors, motors, LED lights, etc).

❖ It has limited processing power, memory and I/O compared to computer.

8

Getting Started

- ❖ 17 groups: each group consists of 4 students (except a 3-student group).
- ❖ Download and install the Arduino software: https://www.arduino.cc/en/Main/Software

Package

- ❖ Each group has 1 Arduino kit. You won't get replacements and have to return it, so don't lose anything!
  - ❖ One kit for each team. Kit contains the following items:
    - – 1 Arduino
    - – 1 USB cable
    - – 1 breadboard
    - – 3 ultrasonic sensors
    - – 4 button switches
    - – Several red, green and yellow LEDs and 1 RGB led
    - – Jumper wires
    - – Resistors

Project 1 Content

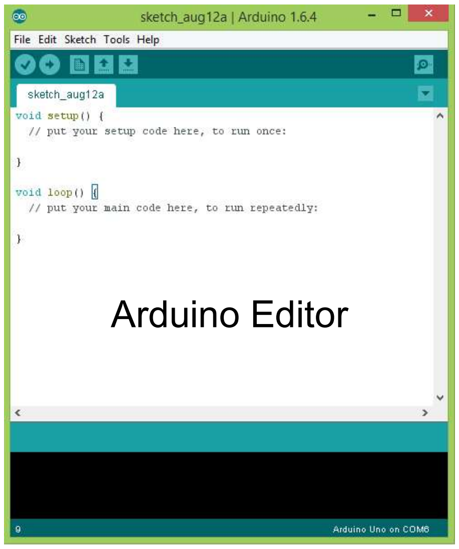
- A simple hands-on example: blink an onboard LED
- Arduino Programming Introduction
- Hands-on 1: Blink LEDs
- Hands-on 2: Control LEDs
- Hands-on 3: Ultrasonic Sensor
- Hands-on 4: Vehicle Controls

Project 1 Content

- A simple hands-on example: blink an onboard LED
- Arduino Programming Introduction
- Hands-on 1: Blink LEDs
- Hands-on 2: Control LEDs
- Hands-on 3: Ultrasonic Sensor
- Hands-on 4: Vehicle Controls

Start Arduino

Arduino Icon



```
void setup() {
  // put your setup code here, to run once:

}

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

}
```

Arduino Editor

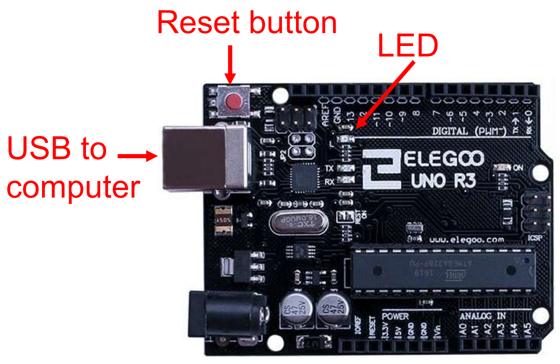
Arduino Code Structure

```
void setup() {
  //setup code to run once after each
  //power-up or reset, e.g., to
  //initialize variables and pin modes

}


void loop() {
  //main code to run repeatedly

}
```

Example Program: Blink Arduino LED

Task: Make an onboard LED blink.

❖ We need the following:



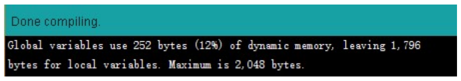
❖ Arduino boards already have an LED connected to pin 13.

Example Program: Blink Arduino LED (cont.)

```
void setup() {
  pinMode(13, OUTPUT);     // initialize digital pin 13
                           // as an output
}

void loop() {
  digitalWrite(13, HIGH); // turn the LED on
                          // (HIGH is the voltage level)
  delay(1000);            // wait for one second
  digitalWrite(13, LOW);  // turn the LED off by making
                          // the voltage LOW
  delay(1000);            // wait for one second
}
```

Compile Arduino Code

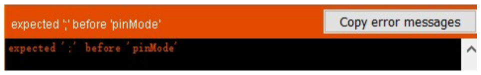
- Save your Arduino code as *blink.*
- Compile your code using ✓
- The code is right when it shows "Done compiling"



```
Done compiling.
Global variables use 252 bytes (12%) of dynamic memory, leaving 1,796
bytes for local variables. Maximum is 2,048 bytes.
```

at the bottom.

- If there's an error in the code, there will be an error message at the bottom, for example:

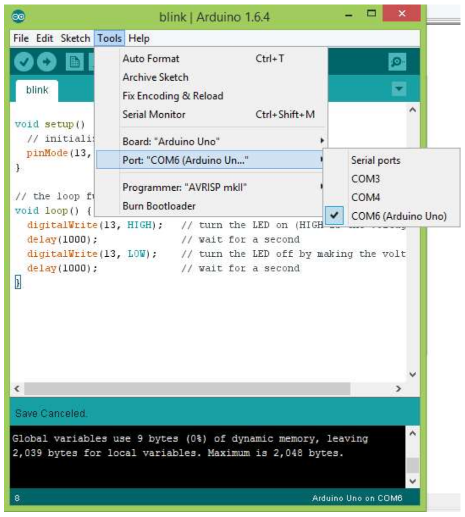


```
expected ';' before 'pinMode'
expected ';' before 'pinMode'
```

- If there're multiple errors in the code, the error message will only show the first error in the code.

Upload and Run the Arduino Code

- Connect the Arduino board to your laptop by plugging it into a USB port.
- Ensure the board is connected by clicking on the *Tools* menu, then check the *Port* sub-menu.
- If the board is not connected, select *COMx (Arduino Uno)* under the *Tools->Port* sub-menu.



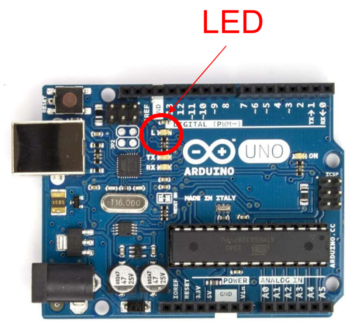
```
void setup()
  // initiali
  pinMode(13,
}

// the loop f
void loop() {
  digitalWrite(13, HIGH);   // turn the LED on (HIGH
  delay(1000);              // wait for a second
  digitalWrite(13, LOW);    // turn the LED off by making the volt
  delay(1000);              // wait for a second
}
```

```
Save Canceled.
Global variables use 9 bytes (0%) of dynamic memory, leaving
2,039 bytes for local variables. Maximum is 2,048 bytes.
```

Upload and Run the Arduino Code (cont.)

❖ Upload the code using

❖ When the upload is successful, you will see the LED blinks on the board every one second.



Project 1 Content

- A simple hands-on example: blink an onboard LED
- Arduino programming Introduction
- Hands-on 1: Blink LEDs
- Hands-on 2: Control LEDs
- Hands-on 3: Ultrasonic Sensor
- Hands-on 4: Vehicle Controls

Arduino Programming

❖ Arduino Program Structure

```
//Define global variables
………
void setup() {
  //setup code to run once after each
  //power-up or reset, such as to
  //initialize variables and pin modes
}

void loop() {
  //main code to run repeatedly

}
```

CU-ICAR CLEMSON UNIVERSITY INTERNATIONAL CENTER FOR AUTOMOTIVE RESEARCH

Variables

- byte x; // 8-bit byte variable: 0 to 255
- char x; // 8-bit character: A, B, C, ….
- int x; // 16-bit integer: -2^15 to to 2^15 -1
- unsigned int x; // 16-bit unsigned integer: 0 to 2^16 -1
- long x; // 32-bit integer: -2^31 to 2^31 -1
- unsigned long x; // 32-bit unsigned integer: 0 to 2^32 -1
- float x; // 32-bit floating-point number: -3.4E+38 to 3.4E+38
- double x; // 64-bit floating-point number: -1.7E+308 to 1.7E+308

CU-ICAR CLEMSON UNIVERSITY INTERNATIONAL CENTER FOR AUTOMOTIVE RESEARCH

Operators

❖ Math. Operators

=	(assignment operator)
+	(addition)
-	(subtraction)
*	(multiplication)
/	(division)
%	(modulo)

Operators

❖ Comparison operators:

==	(equal to)
!=	(not equal to)
<	(less than)
>	(greater than)
<=	(less than or equal to)
>=	(greater than or equal to)

❖ Boolean operators:

&&	(and)
||	(or)
!	(not)

"IF" Statement

- "IF" is used to test whether a certain condition has been reached.
- Example:

```
if (variable > 100)
  // action


if (variable > 100) {
  // action 1
  // action 2
  // …
}
```

25

"IF … ELSE" Statement

- "IF … ELSE" allows multiple tests to be grouped together.
- Example:

```
if (variable > 100)
{
  // action 1
}
else
{
  // action 2
}
```

26

“IF … ELSE” Statement (cont.)

```
if (variable < 100)
{ // action 1
}
else if (variable >= 100 && variable < 200)
{ // action 2
}
else
{  // action 3
}
```

“FOR” Statement

- “FOR” is used to repeat some statements.
- Example:

```
for (int i = 0; i < 10; i++)
{
  // action
}
```

- In this example, the “action” will be executed 10 times.

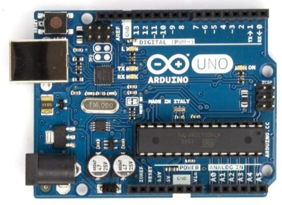
Project 1 Content

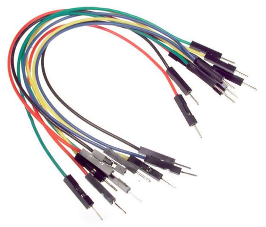
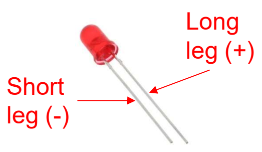
- A simple hands-on example: blink an onboard LED
- Arduino programming Introduction
- Hands-on 1: Blink LEDs
- Hands-on 2: Control LEDs
- Hands-on 3: Ultrasonic Sensor
- Hands-on 4: Vehicle Controls

Try to Blink an External LED

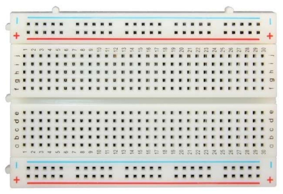
Task: Make an external LED automatically blink.

❖ We need the following:

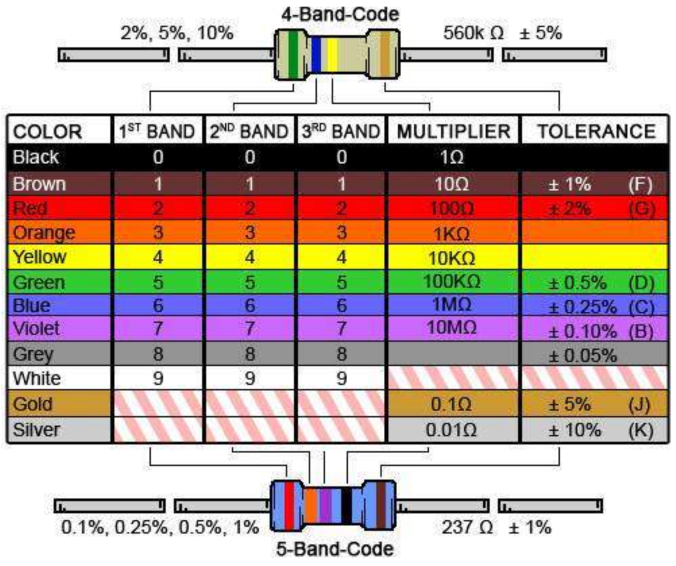


1 kohm



Resistor


4-Band-Code
2%, 5%, 10%
560k Ω ± 5%
0.1%, 0.25%, 0.5%, 1%
237 Ω ± 1%
5-Band-Code

COLOR	1ST BAND	2ND BAND	3RD BAND	MULTIPLIER	TOLERANCE
Black	0	0	0	1Ω	
Brown	1	1	1	10Ω	± 1% (F)
Red	2	2	2	100Ω	± 2% (G)
Orange	3	3	3	1KΩ	
Yellow	4	4	4	10KΩ	
Green	5	5	5	100KΩ	± 0.5% (D)
Blue	6	6	6	1MΩ	± 0.25% (C)
Violet	7	7	7	10MΩ	± 0.10% (B)
Grey	8	8	8		± 0.05%
White	9	9	9	▨	▨
Gold	▨	▨	▨	0.1Ω	± 5% (J)
Silver	▨	▨	▨	0.01Ω	± 10% (K)

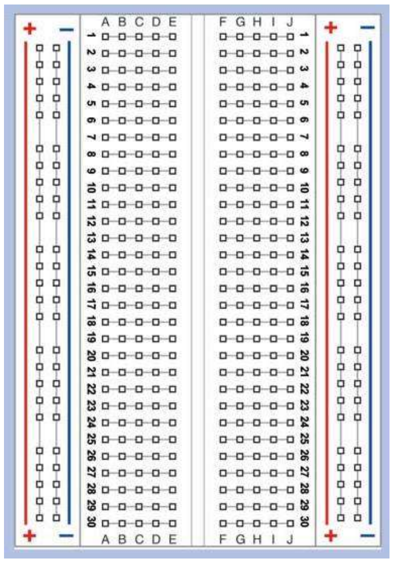
130 ohm

3.3 kohm

1 kohm

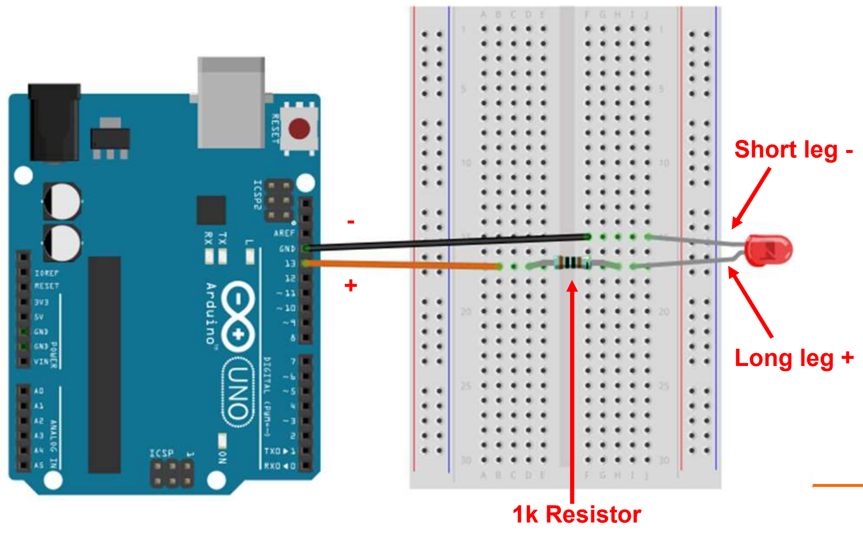
Breadboard

- Breadboard is used to test circuit.
- One of the main advantages is that the components are not soldered and if they are positioned incorrectly they can be moved easily.
- Letters are used to identify vertical columns.
- Numbers to identify horizontal rows.
- Lines show how some vertical columns and horizontal rows are internally connected.
- When power is applied to the breadboard, current flows along these internal connections.



Blink External LED: Connection

IMPORTANT: Power off the board before you do the connection! (Plug out the USB cable!)



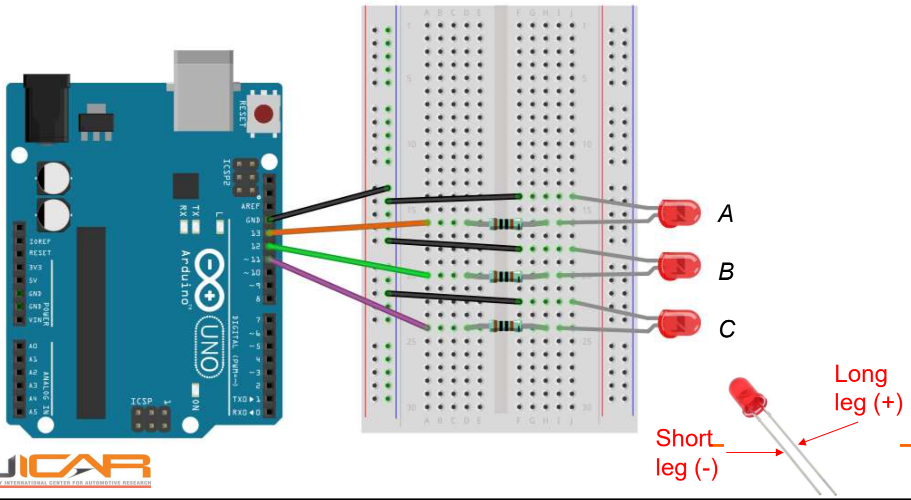
Upload and Run the Arduino Code

- Connect the Arduino board to your laptop by plugging it into a USB port.
- Ensure the board is connected by clicking on the *Tools* menu, then check the *Port* sub-menu.
- If the board is not connected, select *COMx (Arduino Uno)* under the *Port* sub-menu.
- Upload the code.
- Once the compilation has finished the application upload procedure runs.
- When the upload is successful, you will see the LED blinks.

Challenge: Blink 3 LEDs: Connection

Task: Make the LED A, B and C blink sequentially with a time interval of 200 ms.

IMPORTANT: Power off the board before you do the connection!



Blink 3 LEDs: Programming

```
void setup() {
  pinMode(13, OUTPUT);
  pinMode(12, OUTPUT);
  pinMode(11, OUTPUT);
}
void loop() {
  digitalWrite(11, LOW);
  digitalWrite(13, HIGH);
  delay(200);
  digitalWrite(13, LOW);
  digitalWrite(12, HIGH);
  delay(200);
  digitalWrite(12, LOW);
  digitalWrite(11, HIGH);
  delay(200);
}
```

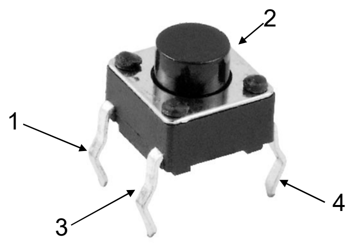
Project 1 Content

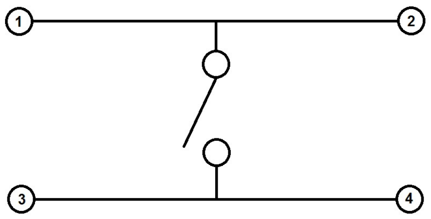
- A simple hands-on example: blink an onboard LED
- Arduino programming Introduction
- Hands-on 1: Blink LEDs
- Hands-on 2: Control LEDs
- Hands-on 3: Ultrasonic Sensor
- Hands-on 4: Vehicle Controls

37

Control LEDs

- In this task, you will learn to use push buttons to turn an LED on and off.
- You will learn to use "if" statement in Arduino.
- Pin 1 and pin 2 are always connected. Pin 3 and pin 4 are always connected.
- When button is pressed down, all four pins are connected.

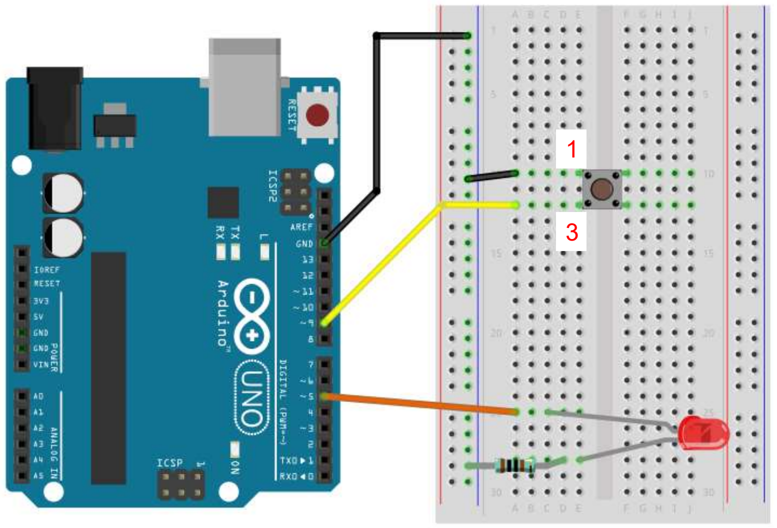


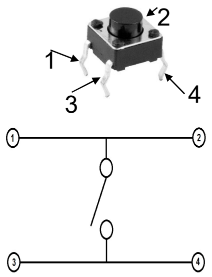


38

Control LED by One Button: Connection

Task: Press once to turn on LED; Press one more time to turn off LED.





Control LED by One Button: Programming

```
int ledPin = 5;          //Assign LED pin
int buttonpin = 9;       //Assign Button A pin
int state = LOW;         //Initialize state
int previous = LOW;      //Define previous state

void setup()
{
  pinMode(ledPin, OUTPUT);
  pinMode(buttonpin, INPUT_PULLUP);
}
```

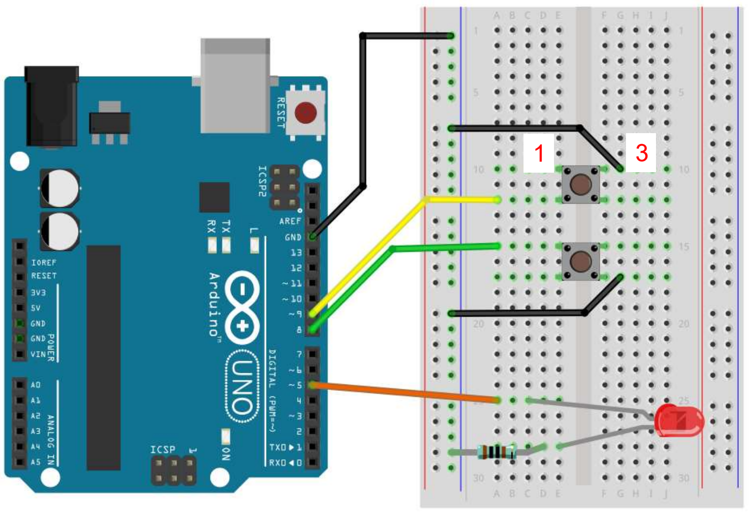
Control LED by One Button: Programming

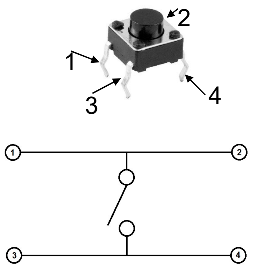
```
void loop()
{
  if (digitalRead(buttonpin) == LOW)
  {
    if (previous == LOW){
      state = HIGH;
    }
    else{
      state = LOW;
    }
    previous=state;
    delay(200);
  }
  digitalWrite(ledPin, state);
}
```

41

Control LED by 2 buttons: Connection

Task: Press one button to turn on LED; Press another button to turn off LED.





42

Control LED by 2 buttons : Programming

```
int ledPin = 5;          //Assign LED pin
int buttonApin = 9;      //Assign Button A pin
int buttonBpin = 8;      //Assign Button B pin

void setup()
{
  pinMode(ledPin, OUTPUT);//Define LED pin as output
  //Define Button pins as INPUT_PULLUP
  pinMode(buttonApin, INPUT_PULLUP);
  pinMode(buttonBpin, INPUT_PULLUP);
  //INPUT_PULLUP, means that if nothing else is
  connected to the input it should be 'pulled up' to
  HIGH.
}
```

43

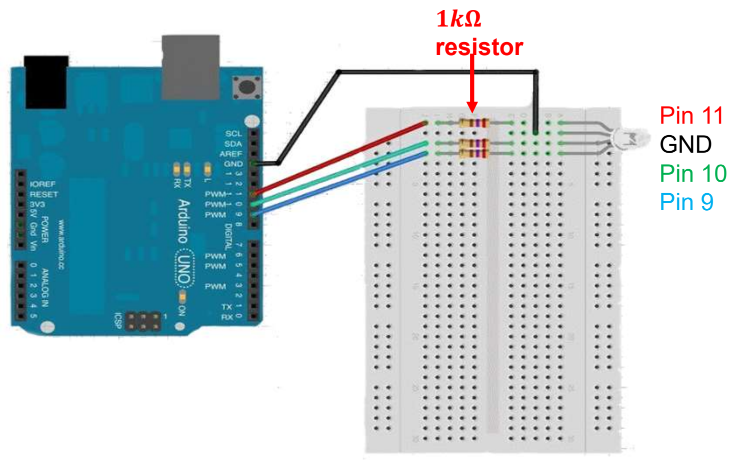
Control LED by 2 buttons: Programming

```
void loop()
{
  //When button A is pressed down, lighten the LED.
  if (digitalRead(buttonApin) == LOW)
  {
    digitalWrite(ledPin, HIGH);
  }
  // When button B is pressed down, turn off the LED.
  if (digitalRead(buttonBpin) == LOW)
  {
    digitalWrite(ledPin, LOW);
  }
}
```

44

Program Arduino to Control RGB LED

❖ Switch LED color between red, green and blue



Contd..

```
int redPin = 11;          //define the red pin
int greenPin = 10;        //define the green pin
int bluePin = 9;          //define the blue pin
int i;                    //define a number for counting

void setup()
{
  pinMode(redPin, OUTPUT);    //define pin as output pin
  pinMode(greenPin, OUTPUT);  //define pin as output pin
  pinMode(bluePin, OUTPUT);   //define pin as output pin

}
```

Contd..

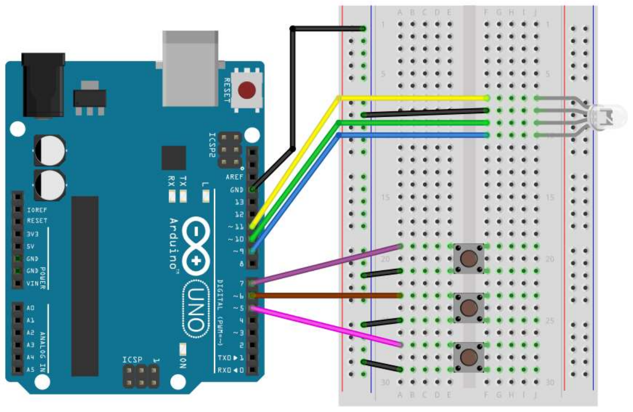
```
void loop()
{
  for (i=1;i<=255;i++)  //loop for color spectrum
  { setColor(i, 0, 0);  //set color (R,G,B)
    delay(10);}         //delay to see output
  for (i=1;i<=255;i++)
  { setColor(255-i, i, 0); //same process
    delay(10);}
  for (i=1;i<=255;i++)
  { setColor(0, 255-i, i);
    delay(10);}
  for (i=1;i<=255;i++)
  { setColor(0, 0, 255-i);
    delay(10);}}
```

Contd..

```
 void setColor(int red, int green, int blue)
{
  analogWrite(redPin, red);
  analogWrite(greenPin, green);
  analogWrite(bluePin, blue);
}
```

Program Arduino to Control RGB LED Using Three Button Switches

- Use three buttons to control the color of the LED



Contd..

```
int redLEDPin = 11;
int greenLEDPin = 10;
int blueLEDPin = 9;

int redSwitchPin = 7;
int greenSwitchPin = 6;
int blueSwitchPin = 5;

int red = 0;
int blue = 0;
int green = 0;
```

Contd..

```
void setup()
{
  pinMode(redLEDPin, OUTPUT);
  pinMode(greenLEDPin, OUTPUT);
  pinMode(blueLEDPin, OUTPUT);
  pinMode(redSwitchPin, INPUT_PULLUP);
  pinMode(greenSwitchPin, INPUT_PULLUP);
  pinMode(blueSwitchPin, INPUT_PULLUP);
}
void loop()
{
  if (digitalRead(redSwitchPin) == LOW)
  {
    red ++;
    if (red > 255) red = 0;
  }
```

Contd..

```
if (digitalRead(greenSwitchPin) == LOW)
  {
    green ++;
    if (green > 255) green = 0;
  }
  if (digitalRead(blueSwitchPin) == LOW)
  {
    blue ++;
    if (blue > 255) blue = 0;
  }
  analogWrite(redLEDPin, red);
  analogWrite(greenLEDPin, green);
  analogWrite(blueLEDPin, blue);
  delay(10);
}
```

Arduino Programming

- More on Arduino programming information can be learned from:

  https://www.Arduino.cc/en/Reference/HomePage