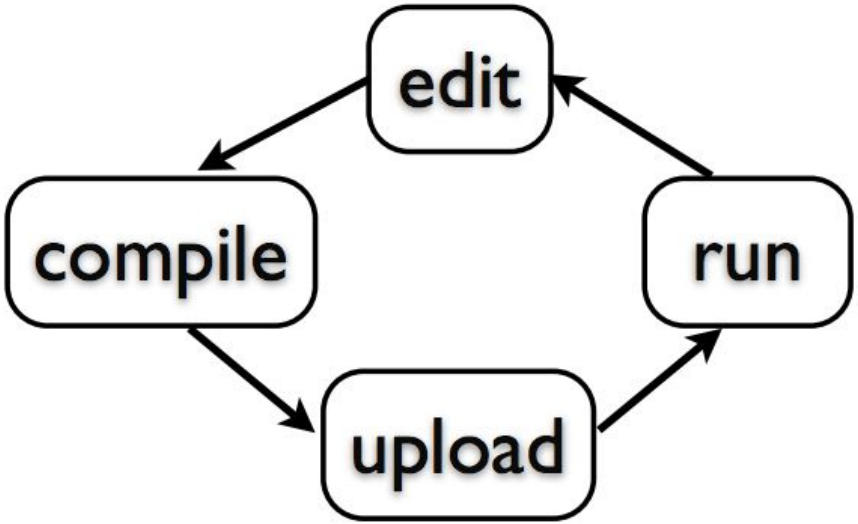
Intro to Arduino- 2

Maya Nasr
MIT-IIT Robotics Program 2017

Development Cycle

Edit → compile → upload → run



Yesterday, we used these functions in our RX/TX Blink Lab

- **`pinMode(pin, mode)`** – set a pin as `INPUT` or `OUTPUT`
- **`digitalWrite(pin, value)`** – set a digital pin that is set as an `OUTPUT` as either `HIGH` (pulled to +5 volts) or `LOW` (pulled to ground).
- **`delay()`** – wait an amount of time

What is a Servo?

- Servo motors are an easy way to add motion to your electronics projects.

What is a Servo?

- Servo motors are an easy way to add motion to your electronics projects.
- They're useful because you can instruct these small motors how far to turn, and they do it for you.

What is a Servo?

- Servo motors are an easy way to add motion to your electronics projects.
- They're useful because you can instruct these small motors how far to turn, and they do it for you.
- In a servo motor you will find a small DC motor, a potentiometer, gear arrangement and an intelligent circuitry.

What is a Servo?

- Servo motors are an easy way to add motion to your electronics projects.
- They're useful because you can instruct these small motors how far to turn, and they do it for you.
- In a servo motor you will find a small DC motor, a potentiometer, gear arrangement and an intelligent circuitry.
- The intelligent circuitry along with the potentiometer makes the servo to rotate according to our wishes.

https://www.youtube.com/watch?v=bu3SPwzcocU

Servo Circuit

- Servo motors have three wires: power, ground, and signal.

Servo Circuit

- Servo motors have three wires: power, ground, and signal.
- The power wire is the red wire, and should be connected to the 5V pin on the Arduino board.

Servo Circuit

- Servo motors have three wires: power, ground, and signal.
- The power wire is the red wire, and should be connected to the 5V pin on the Arduino board.
- The ground wire is the brown and should be connected to a ground pin on the Arduino board.

Servo Circuit

- Servo motors have three wires: power, ground, and signal.
- The power wire is the red wire, and should be connected to the 5V pin on the Arduino board.
- The ground wire is the brown and should be connected to a ground pin on the Arduino board.
- The signal pin is the orange wire and should be connected to a digital pin on the Arduino board.

attach()

Description

Attach the Servo variable to a pin.

attach()

Description

Attach the Servo variable to a pin.

Syntax

servo.attach(pin)

attach()

Description

Attach the Servo variable to a pin.

Syntax

servo.attach(pin)

Parameters

servo: a variable of type Servo

pin: the number of the pin that the servo is attached to

write()

Description

Writes a value to the servo, controlling the shaft accordingly. This will set the angle of the shaft (in degrees), moving the shaft to that orientation.

write()

Description

Writes a value to the servo, controlling the shaft accordingly. This will set the angle of the shaft (in degrees), moving the shaft to that orientation.

Syntax

servo.write(angle)

write()

Description

Writes a value to the servo, controlling the shaft accordingly. This will set the angle of the shaft (in degrees), moving the shaft to that orientation.

Syntax

servo.write(angle)

Parameters

servo: a variable of type Servo

angle: the value to write to the servo, from 0 to 180

Servo library

This library allows an Arduino board to control servo motors.

servo_test

```
// Include the Servo library
#include <Servo.h>
```

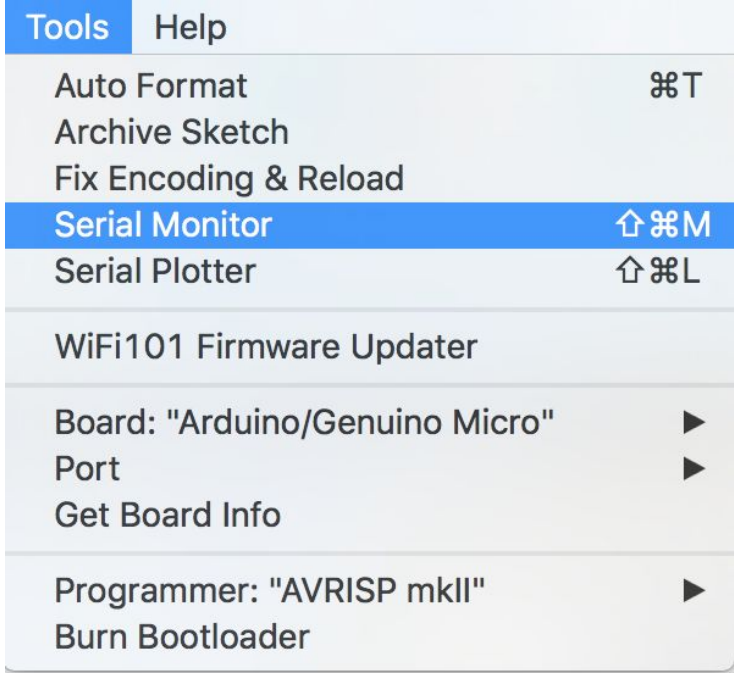
Servo Code

servo_test

```
// Include the Servo library
#include <Servo.h>
// Declare the Servo pin
int yServoPin = 2;
int xServoPin = 3;
// Create a servo object
Servo ServoX, ServoY;
int mn = 60, mx = 120;
void setup() {
   // We need to attach the servo to the used pin number
   ServoX.attach(xServoPin);
   ServoY.attach(yServoPin);
}
void loop(){
   // Make servo go to 0 degrees
   for(int i=mn ; i < mx ; ++i){
     ServoX.write(i);  // tell servo to go to position in variable 'i'
     ServoY.write(i);  // tell servo to go to position in variable 'i'
     delay(40);  // waits 40ms for the servo to reach the position
   }

   for(int i=mx ; i > mn ; --i){
     ServoX.write(i);
     ServoY.write(i);
     delay(40);
   }
}
```

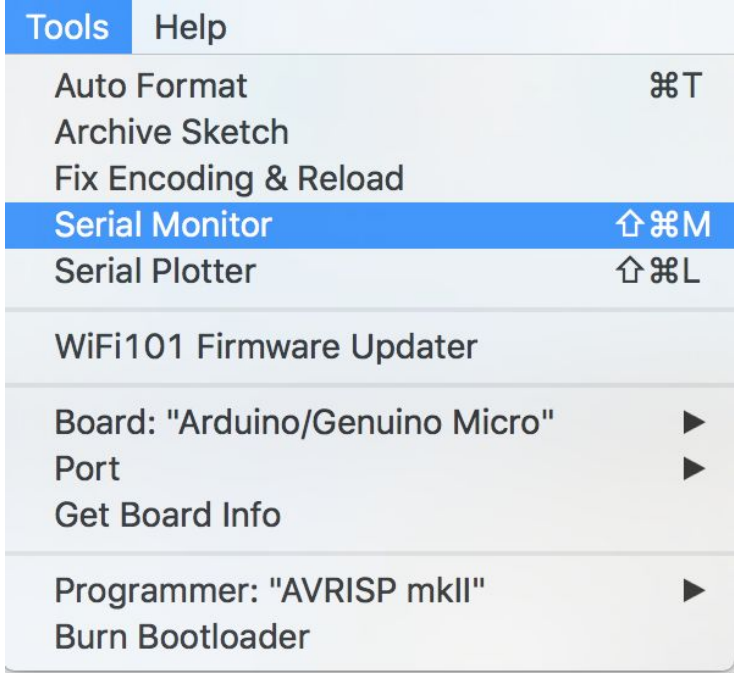
Serial Monitor

- The Arduino IDE has a feature that can be a great help in debugging sketches or controlling Arduino from your computer's keyboard.



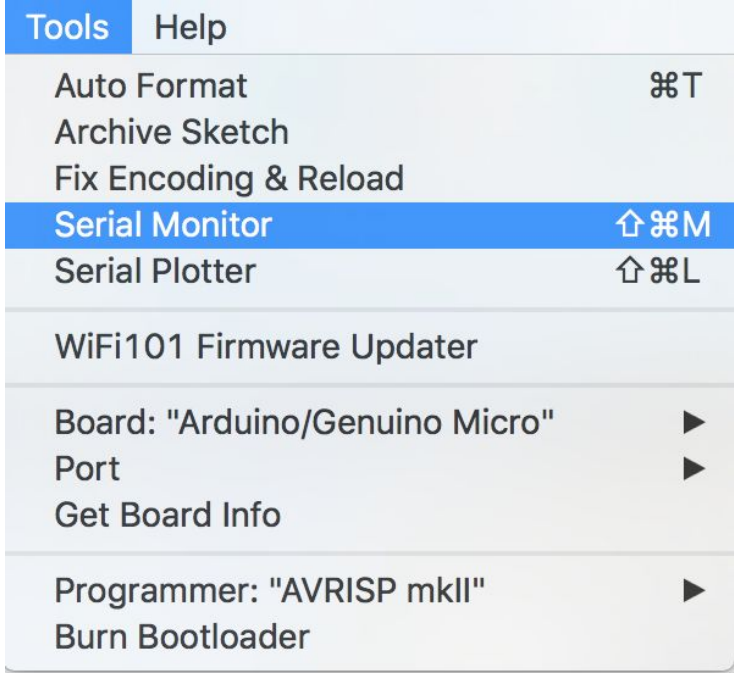
Serial Monitor

- The Arduino IDE has a feature that can be a great help in debugging sketches or controlling Arduino from your computer's keyboard.
- The Serial Monitor is a separate pop-up window that acts as a separate terminal that communicates by receiving and sending Serial Data.



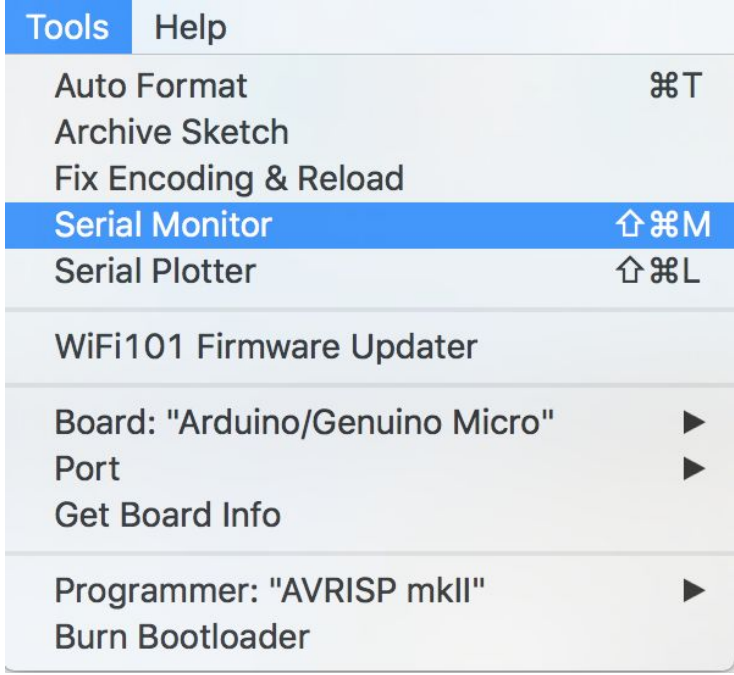
Serial Monitor

- The Arduino IDE has a feature that can be a great help in debugging sketches or controlling Arduino from your computer's keyboard.
- The Serial Monitor is a separate pop-up window that acts as a separate terminal that communicates by receiving and sending Serial Data.
- Serial is used for communication between the Arduino board and a computer or other devices.



Serial Monitor

- The Arduino IDE has a feature that can be a great help in debugging sketches or controlling Arduino from your computer's keyboard.
- The Serial Monitor is a separate pop-up window that acts as a separate terminal that communicates by receiving and sending Serial Data.
- Serial is used for communication between the Arduino board and a computer or other devices.
- Serial Data is sent over a single wire (but usually travels over USB in our case) and consists of a series of 1's and 0's sent over the wire.



Serial Monitor

- **Serial.begin(speed)**

  Sets the data rate in bits per second for serial data transmission. For communicating with the computer use 9600.

```
void setup(){
  Serial.begin(9600);
```

Serial Monitor

- **Serial.begin(speed)**

  Sets the data rate in bits per second for serial data transmission. For communicating with the computer use 9600.

- **Serial.println(val)**

  Prints data to the serial port as human-readable text .

```
void setup(){
  Serial.begin(9600);
```

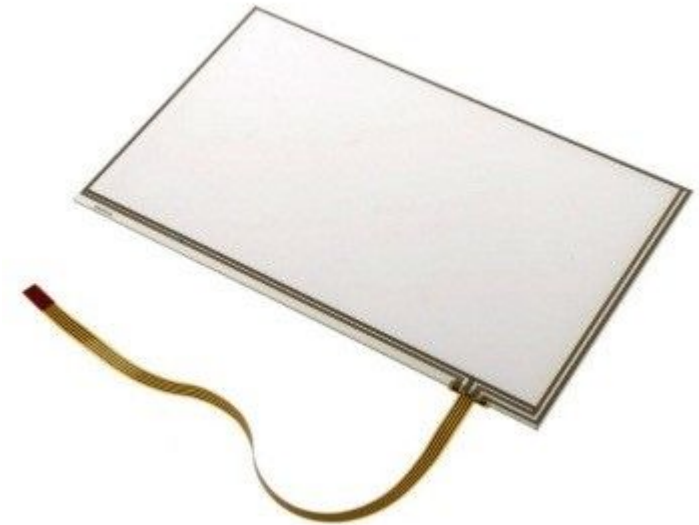
```
int analogValue = 0;    // variable to

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

void loop() {
  // read the analog input on pin 0:
  analogValue = analogRead(0);

  // print it out in many formats:
  Serial.println(analogValue);       /
```

Serial Monitor Lab

Print "Hello World" to the Serial Monitor

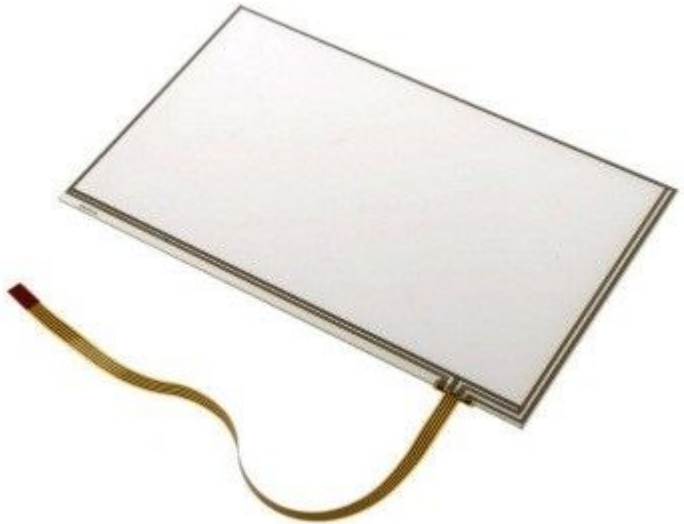
Touchscreen

- Resistive touchscreen displays are composed of multiple layers that are separated by thin spaces.



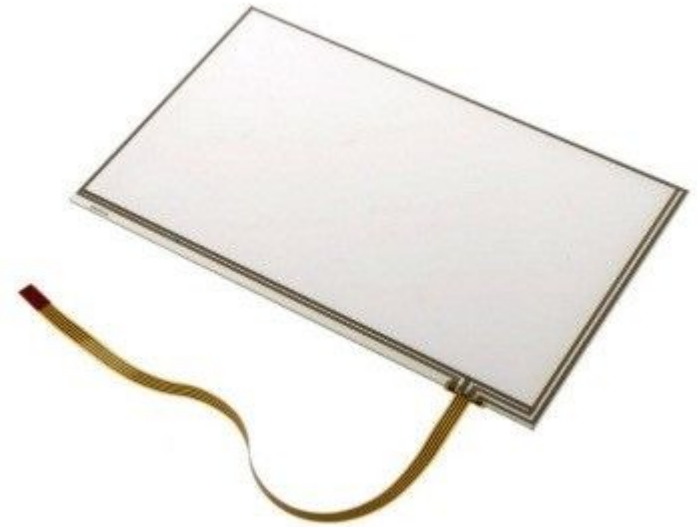
Touchscreen

- Resistive touchscreen displays are composed of multiple layers that are separated by thin spaces.
- Pressure applied to the surface of the display by a finger or ball causes the layers to touch, which completes electrical circuits and tells the device where the user is touching.



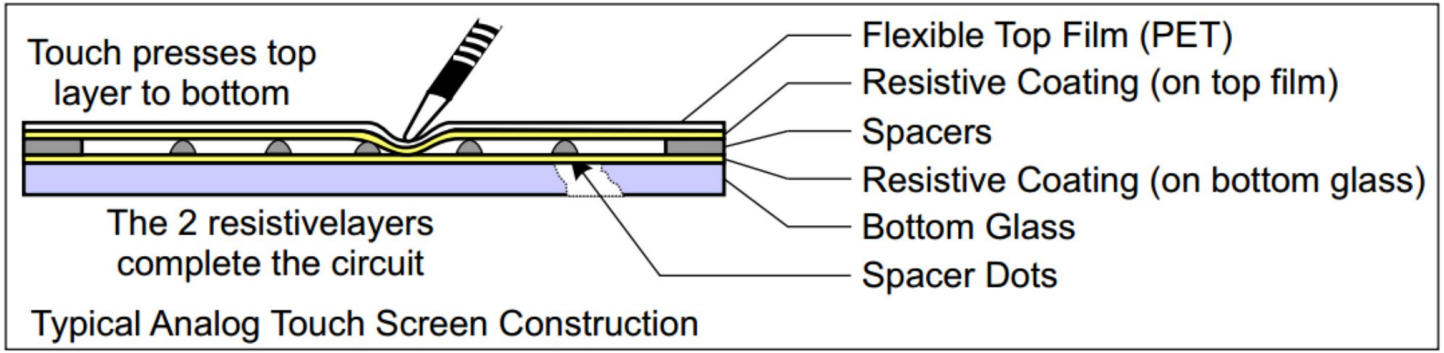
Touchscreen

- Resistive touchscreen displays are composed of multiple layers that are separated by thin spaces.
- Pressure applied to the surface of the display by a finger or ball causes the layers to touch, which completes electrical circuits and tells the device where the user is touching.
- Find out the **X and Y coordinates** of the current point being touched.



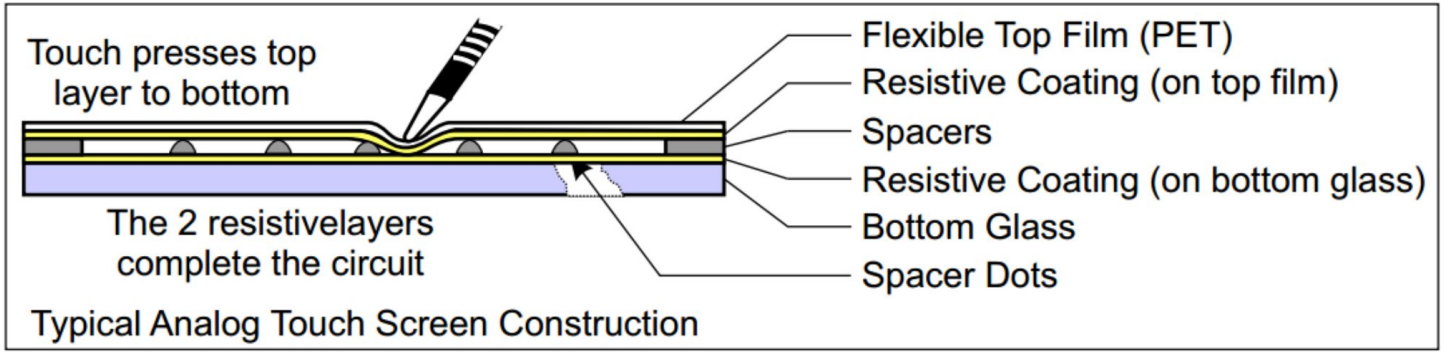
Touchscreen: How does it exactly work?

- A resistive touch screen is constructed with two transparent layers coated with a conductive material stacked on top of each other.



Touchscreen: How does it exactly work?

- A resistive touch screen is constructed with two transparent layers coated with a conductive material stacked on top of each other.



- When pressure is applied by a finger or a stylus on the screen, the top layer makes contact with the lower layer. When a voltage is applied across one of the layers, a voltage divider is created.

Touchscreen: How does it exactly work?

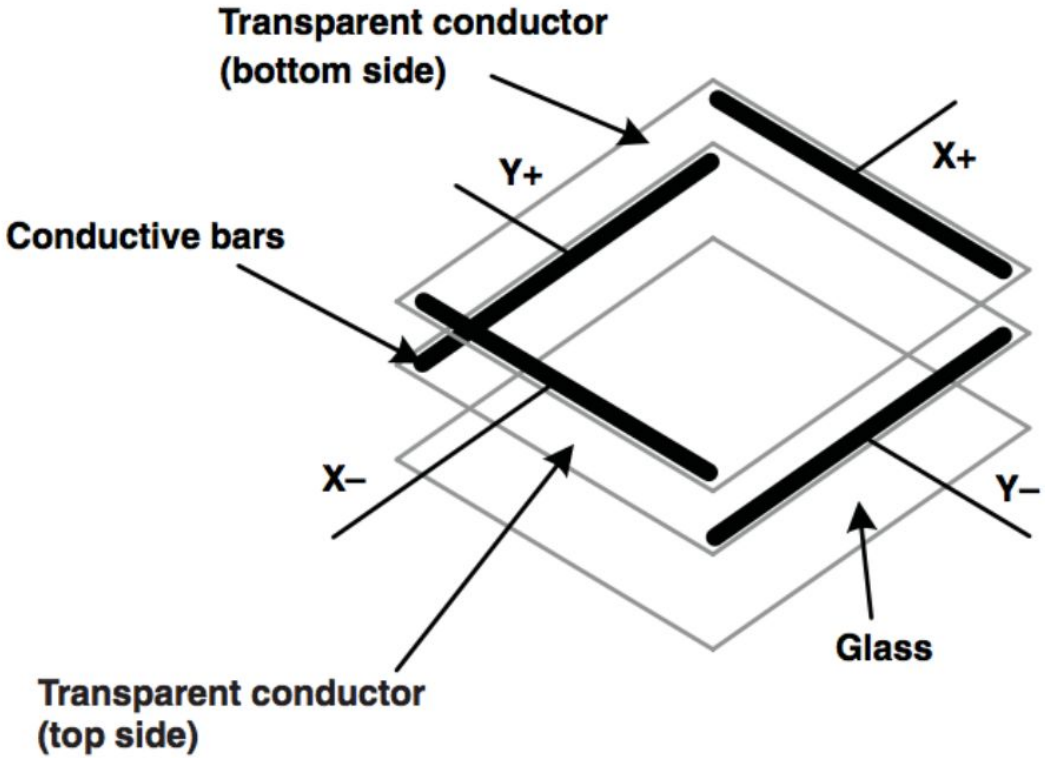
- The coordinates of a touch can be found by applying a voltage across one layer in the Y direction and reading the voltage created by the voltage divider to find the Y coordinate

Touchscreen: How does it exactly work?

- The coordinates of a touch can be found by applying a voltage across one layer in the Y direction and reading the voltage created by the voltage divider to find the Y coordinate

- Then applying a voltage across the other layer in the X direction and reading the voltage created by the voltage divider to find the X coordinate.

4-Wire TouchScreen



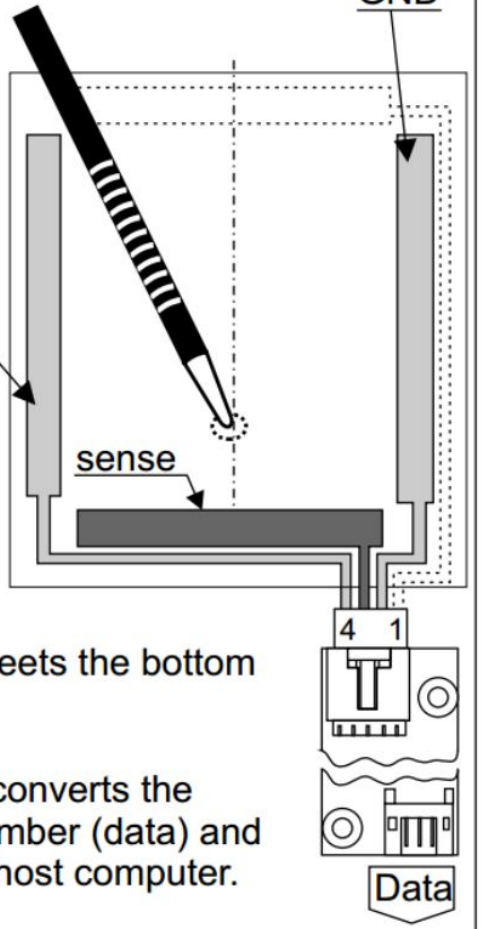
Capturing the "X" Touch

To get the "X" touch position, the controller sets Pin4 to +5V and Pin2 to GND (0V).

Pin1 is left unconnected.

The controller uses Pin3 to read the voltage where the top layer meets the bottom layer.

The controller converts the voltage to a number (data) and sends it to the host computer.



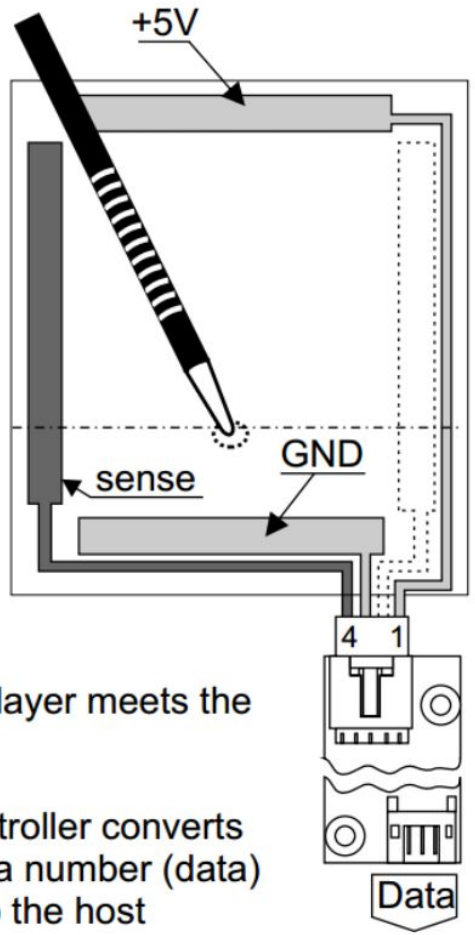
Capturing the "Y" Touch

To get the "Y" touch position the controller sets Pin1 to +5V and Pin3 to GND (0V)

Pin2 is left unconnected.

The controller uses Pin4 to read the voltage where the top layer meets the bottom layer.

Again, the controller converts the voltage to a number (data) and sends it to the host computer.

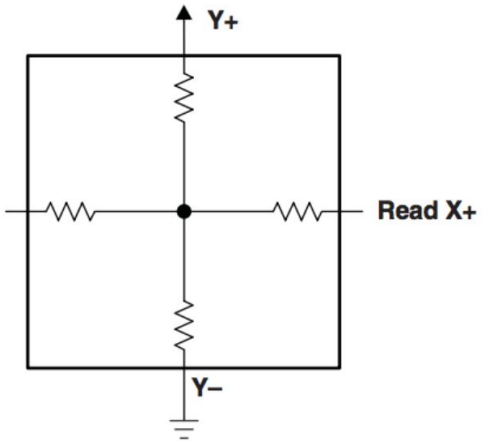


rogram 2017

4-Wire TouchScreen

- The x and y coordinates of a touch on a 4-wire touch screen can be read in two steps.

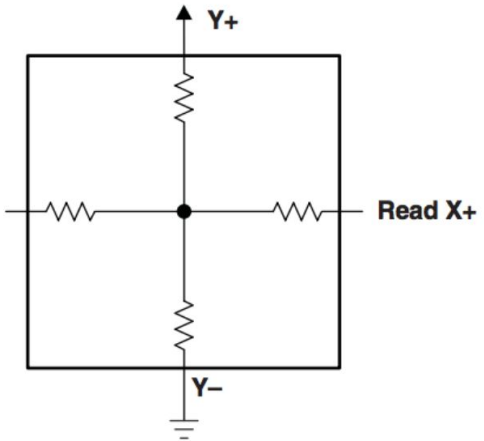
$$y = \frac{V_{X+}}{V_{Drive}} \times height_{screen}$$



4-Wire TouchScreen

- The x and y coordinates of a touch on a 4-wire touch screen can be read in two steps.
- First, Y+ is driven high, Y– is driven to ground, and the voltage at X+ is measured.

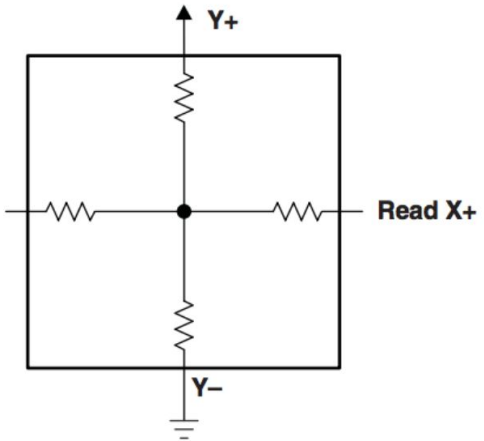
$$y = \frac{V_{X+}}{V_{Drive}} \times height_{screen}$$



4-Wire TouchScreen

- The x and y coordinates of a touch on a 4-wire touch screen can be read in two steps.
- First, Y+ is driven high, Y– is driven to ground, and the voltage at X+ is measured.
- The ratio of this measured voltage to the drive voltage applied is equal to the ratio of the y coordinate to the height of the touch screen.

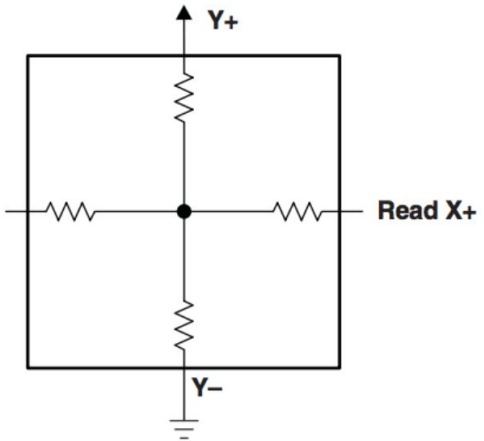
$$y = \frac{V_{X+}}{V_{Drive}} \times height_{screen}$$



4-Wire TouchScreen

- The x and y coordinates of a touch on a 4-wire touch screen can be read in two steps.
- First, Y+ is driven high, Y– is driven to ground, and the voltage at X+ is measured.
- The ratio of this measured voltage to the drive voltage applied is equal to the ratio of the y coordinate to the height of the touch screen.
- The y coordinate can be calculated.

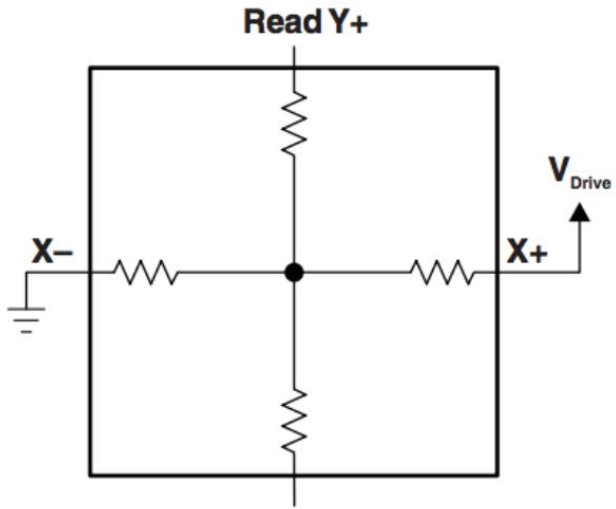
$$y = \frac{V_{X+}}{V_{Drive}} \times height_{screen}$$



4-Wire TouchScreen

- The x coordinate can be similarly obtained by driving X+ high, driving X– to ground, and measuring the voltage at Y+.

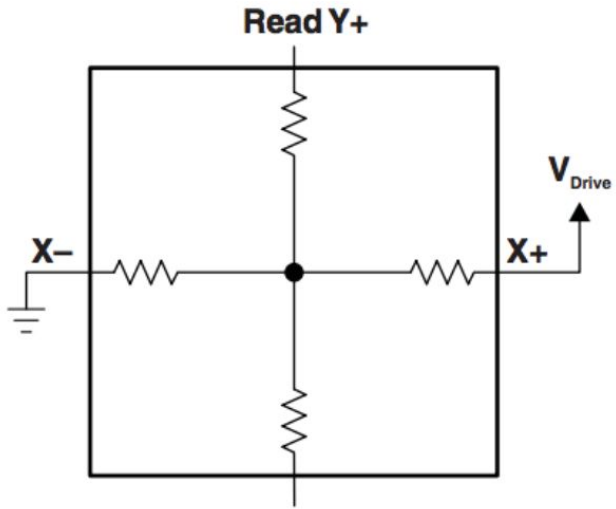
$$x = \frac{V_{Y+}}{V_{Drive}} \times width_{screen}$$



4-Wire TouchScreen

$$x = \frac{V_{Y+}}{V_{Drive}} \times width_{screen}$$

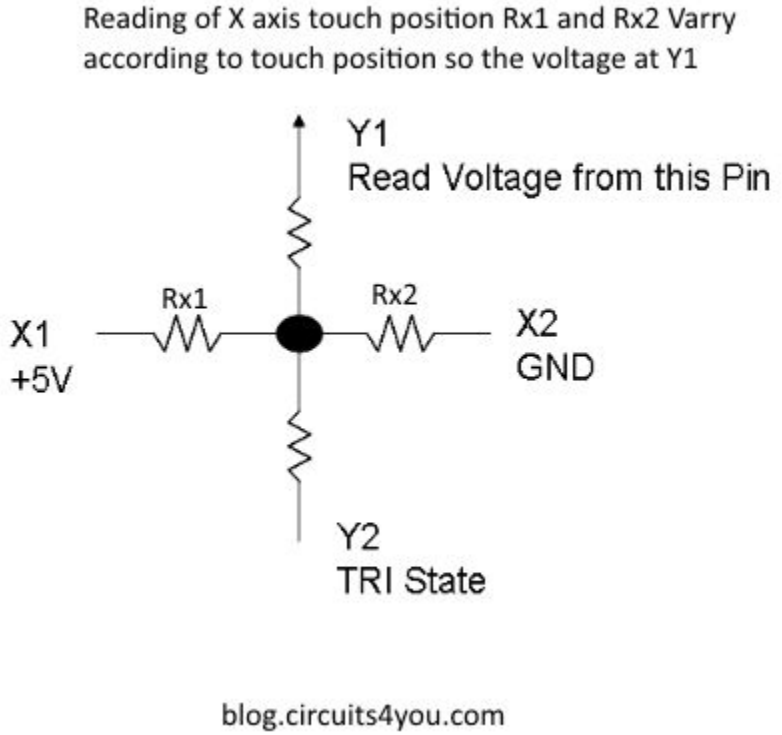
- The x coordinate can be similarly obtained by driving X+ high, driving X– to ground, and measuring the voltage at Y+.
- The ratio of this measured voltage to the drive voltage applied is equal to the ratio of the x coordinate to the width of the touch screen.



Measure X axis Voltage

To measure X axis voltage

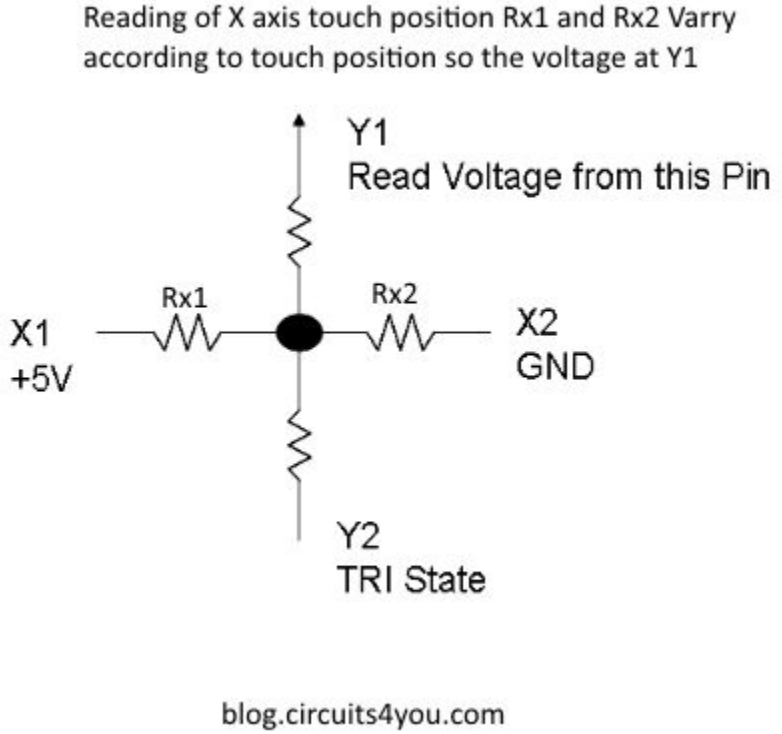
a. We are going to measure voltage on Y1



Measure X axis Voltage

To measure X axis voltage

a. We are going to measure voltage on Y1
-> set Y1 pin as INPUT

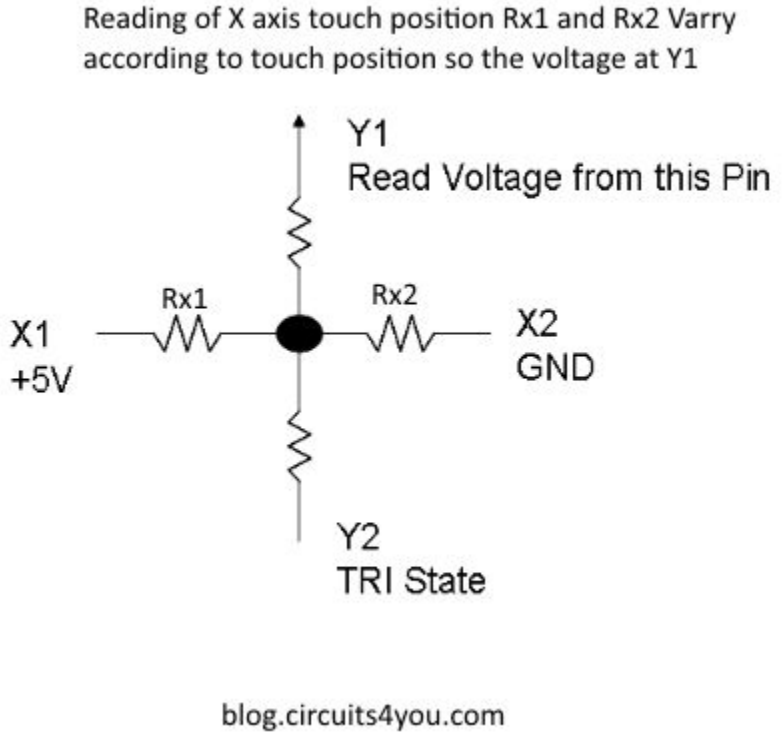


Measure X axis Voltage

To measure X axis voltage

a. We are going to measure voltage on Y1
-> set Y1 pin as INPUT

b. Make Y2 Tristate (remove its influence from circuit)

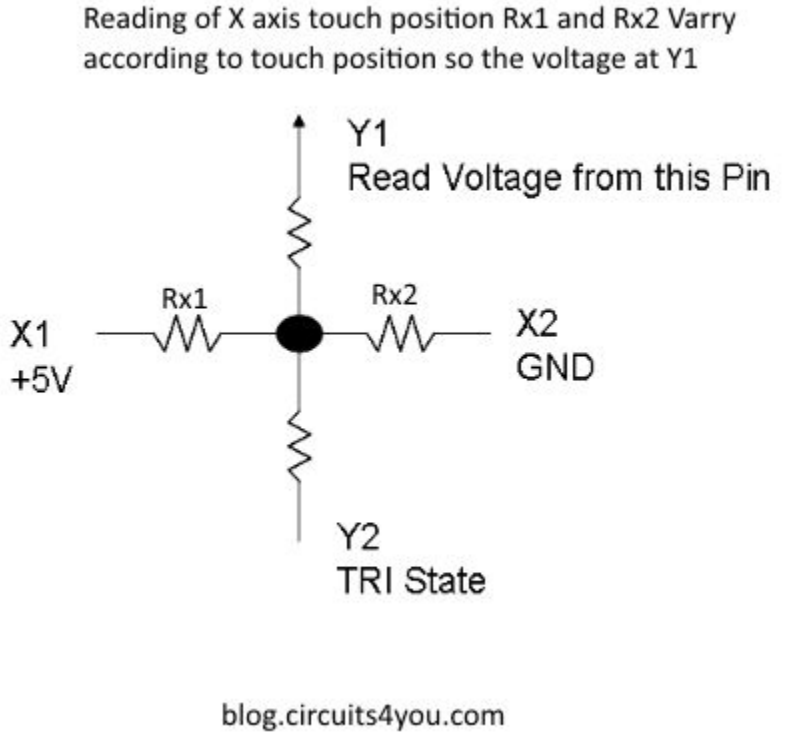


Measure X axis Voltage

To measure X axis voltage

a. We are going to measure voltage on Y1
-> set Y1 pin as INPUT

b. Make Y2 Tristate (remove its influence from circuit)
-> set Y2 as INPUT but LOW



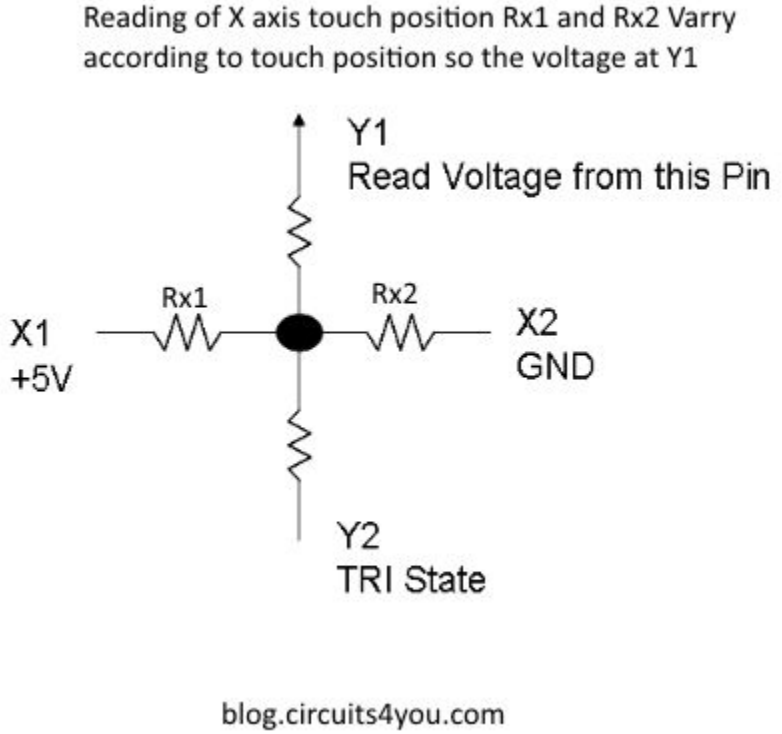
Measure X axis Voltage

To measure X axis voltage

a. We are going to measure voltage on Y1
-> set Y1 pin as INPUT

b. Make Y2 Tristate (remove its influence from circuit)
-> set Y2 as INPUT but LOW

c. Form a voltage divider in X1(+5V) and X2(GND)



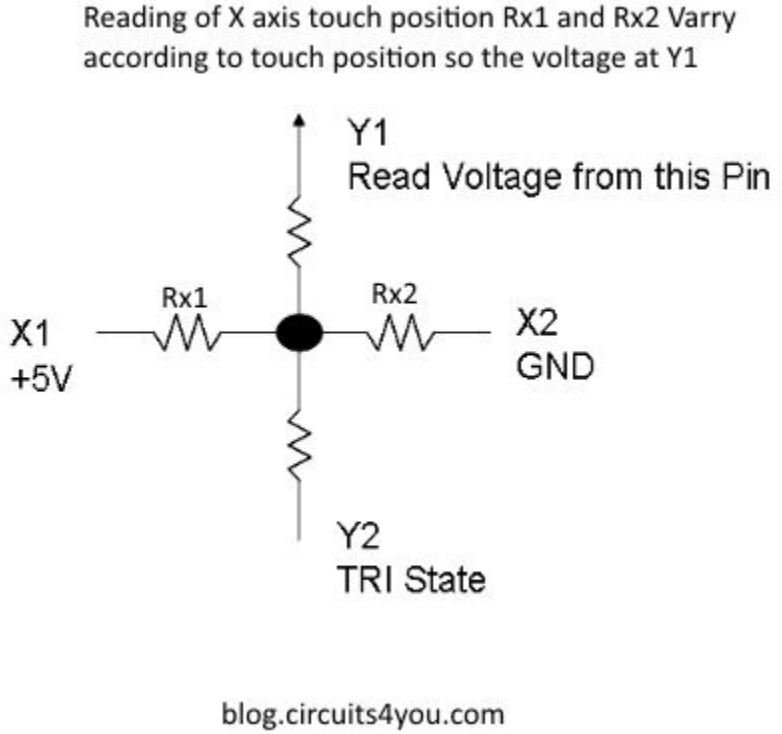
Measure X axis Voltage

To measure X axis voltage

a. We are going to measure voltage on Y1
-> set Y1 pin as INPUT

b. Make Y2 Tristate (remove its influence from circuit)
-> set Y2 as INPUT but LOW

c. Form a voltage divider in X1(+5V) and X2(GND)
-> set X1 as OUTPUT but HIGH
set X2 as OUTPUT but LOW



Measure X axis Voltage

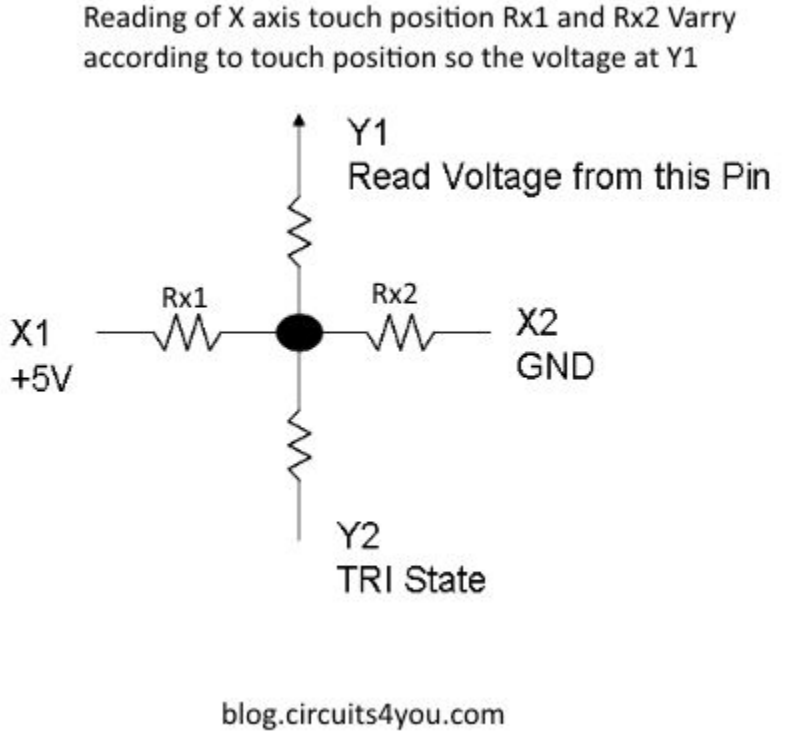
To measure X axis voltage

a. We are going to measure voltage on Y1
-> set Y1 pin as INPUT

b. Make Y2 Tristate (remove its influence from circuit)
-> set Y2 as INPUT but LOW

c. Form a voltage divider in X1(+5V) and X2(GND)
-> set X1 as OUTPUT but HIGH
set X2 as OUTPUT but LOW

d. Read the ADC from Y1 pin (analogRead)



Similarly Measure Y axis Voltage

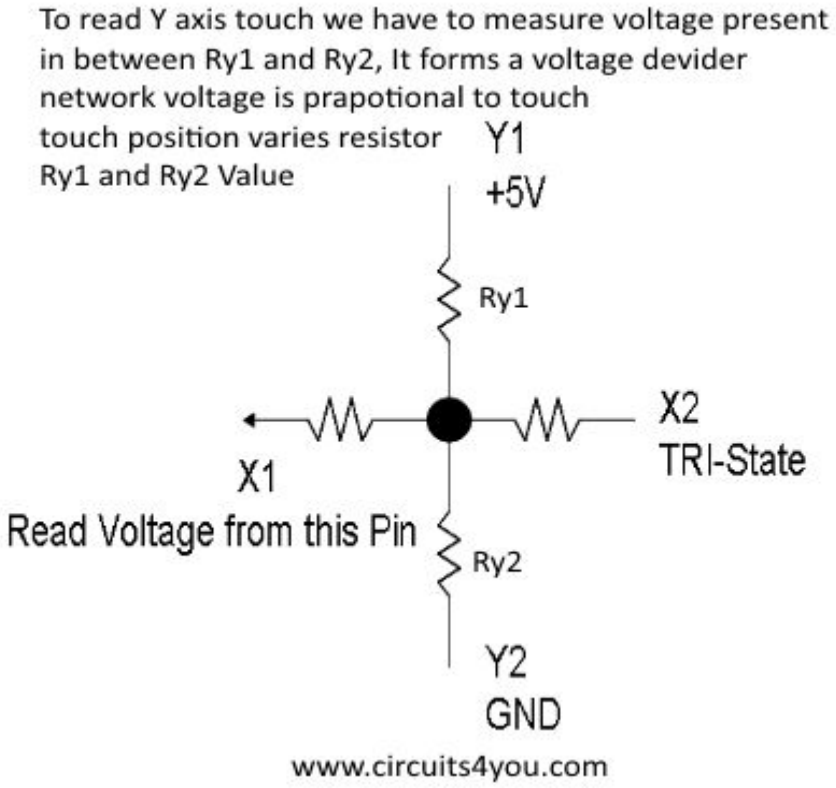
To measure Y axis voltage

a. We are going to measure voltage on X1

b. Make X2 Tristate (remove its influence from circuit)

c. Form a voltage divider in Y1(+5V) and Y2(GND)

d. Read the ADC from X1 pin (analogRead)



Touchscreen Arduino Code

Write an arduino code that gives the X and Y coordinates of a touch point.

Don't forget to:

- Define your Touch screen connection: (Y+ is A0, X+ is A1, Y- is A2 and X- is A3)
- Define your screen resolution: (Xresolution is 740 and Yresolution is 645)