Robot Programming and Control

Part III

TOPICS:

- **Random () function**
- **Getting input from Serial Monitor and getting an output such as turn an LED or speaker on**
- **Flex Sensor and using values to turn on multiple LEDs (or sound)**
- **Shift Registers**

The serial port – sending a random number using the random function

```
void setup(){
  Serial.begin(9600);      //Begin Serial Communication
 }

 void loop(){
  Serial.println(random(1,1000)); //Send Random # to computer
  delay(200);                 //Delay 200ms between values.
 }
```

Receive a serial command from Serial Monitor screen and perform a task such as turning and LED or speaker ON. Any number will turn it ON. Entering 0 will turn it OFF

```
// The onboard LED is on pin # 13 and turns ON when a number is entered via serial port
int onboardLED = 13;

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

 //Set the onboard LED to OUTPUT
 pinMode(onboardLED, OUTPUT);
}

void loop(){
 /* Read serial port, if the number is 0, then turn off LED
 if the number is 1 or greater, turn the LED on. */
 while (Serial.available() > 0) {
 int num=Serial.read()-'0';
 if(num<1){
 digitalWrite(onboardLED, LOW); //Turn Off LED
 } else{
 digitalWrite(onboardLED, HIGH); //Turn On LED
 }
 }
}
```

Receive a serial command from Serial Monitor screen and perform a task such as turning and LED or speaker ON

```
/* Receive a serial command from serial monitor and turn on/off 1-9 LEDs, 0 turns it off */

void setup() {
 // initialize the digital pins as an output.
 pinMode(2, OUTPUT);
 pinMode(3, OUTPUT);
 pinMode(4, OUTPUT);
 pinMode(5, OUTPUT);
 pinMode(6, OUTPUT);
 pinMode(7, OUTPUT);
 pinMode(8, OUTPUT);
 pinMode(9, OUTPUT);
 pinMode(10, OUTPUT);

Serial.begin(9600);
}

void loop() {
 byte byteRead;
```

```
/* check if data has been sent from the computer: */
 if (Serial.available()) {
  /* read the most recent byte */
 byteRead = Serial.read();
 //You have to subtract '0' from the read Byte to convert from text to a number.
 byteRead=byteRead-'0';

 //Turn off all LEDs if the byte Read = 0
 if(byteRead==0){
 //Turn off all LEDS
 digitalWrite(2, LOW);
 digitalWrite(3, LOW);
 digitalWrite(4, LOW);
 digitalWrite(5, LOW);
 digitalWrite(6, LOW);
 digitalWrite(7, LOW);
 digitalWrite(8, LOW);
 digitalWrite(9, LOW);
 digitalWrite(10, LOW);
 }
 //Turn LED ON depending on the byte Read.
if(byteRead>0){
 digitalWrite((byteRead), HIGH); // set the LED on
 } } }
```

Using a bend / flex sensor (or any other analog sensor like a CDS photocell) to control what LEDS get to turn ON

<u>New commands:</u> constrain and map

constrain (x, a, b)

a- the number to constrain, all data types

b- lower end of range, all data types

c- upper end of range, all data types

Example:

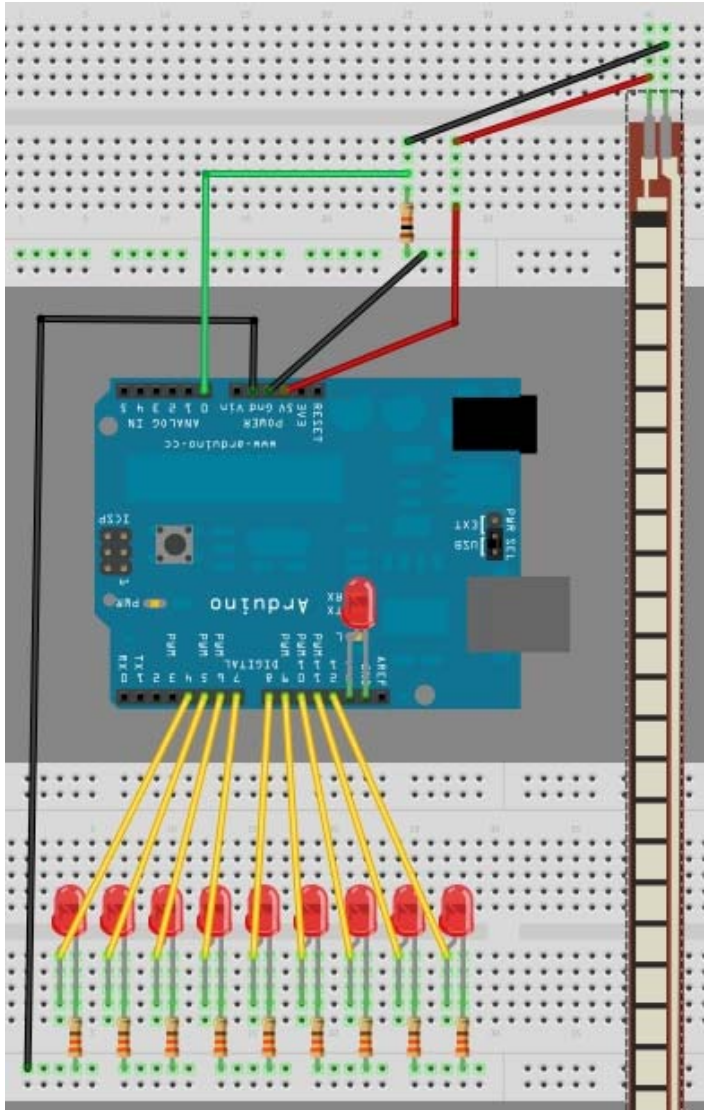
sensVal = constrain(sensVal, 10, 150);

// limits range of sensor values to between 10 and 150

map(value, fromLow, fromHigh, toLow, toHigh)

Example:

val = map(val, 0, 1023, 0, 255);



map(value, fromLow, fromHigh, toLow, toHigh)

```
/* Map an analog value to 8 bits (0 to 255) */
void setup() {}

void loop()
{
  int val = analogRead(0);
  val = map(val, 0, 1023, 0, 255);
  analogWrite(9, val);
}
```

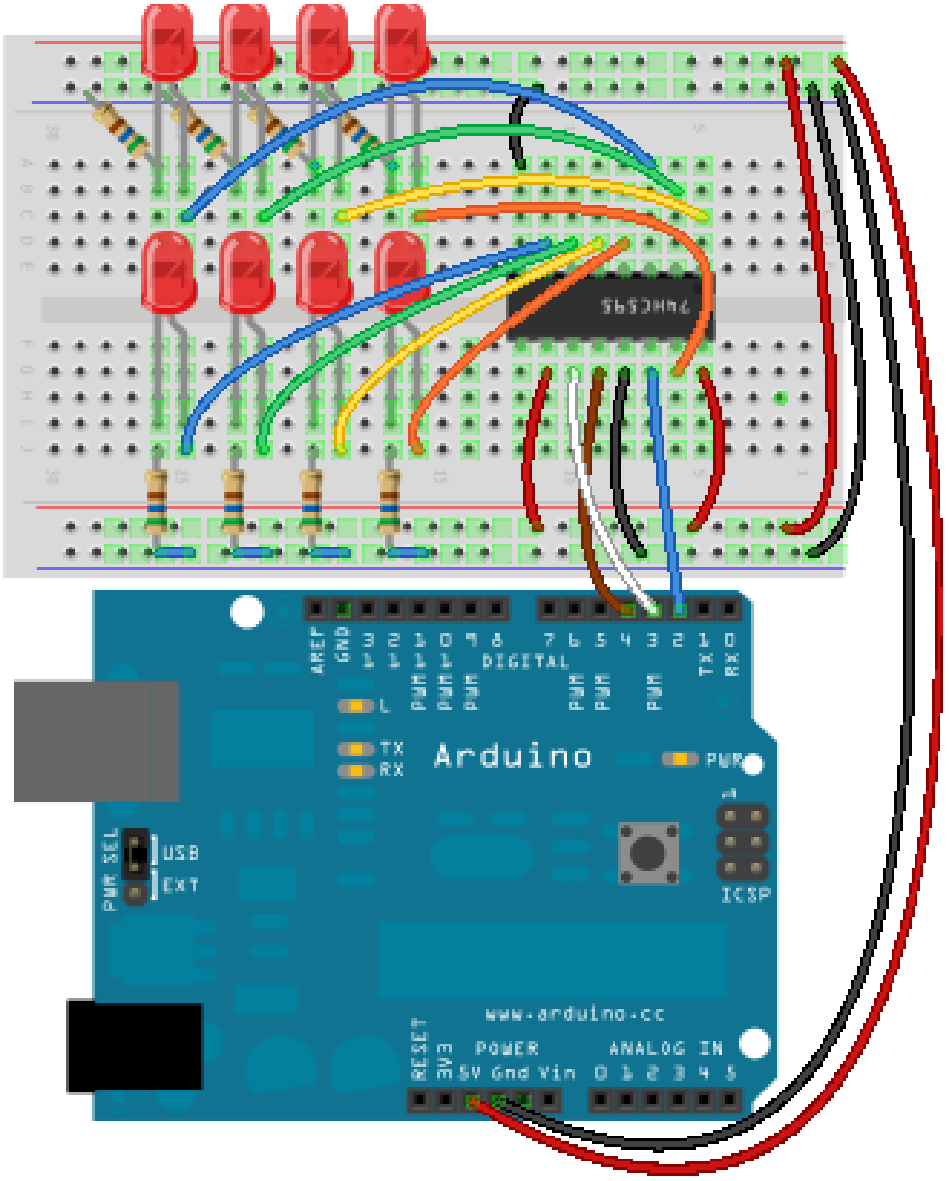
```
//Flex Sensor Pin (flexPin)
//the analog pin the Flex Sensor is connected to
int flexPin = 0;

void setup() {
  for (int i=4; i<14; i++){
    pinMode(i, OUTPUT); //sets the led pins 4 to 13 to output
  }
}

void loop(){
 //Ensure to turn off ALL LEDs before continuing
 for (int i=4; i<14; i++){
    digitalWrite(i, LOW);
  }

 /* Read the flex Level   Adjust the value 130 to 275 to span 4 to 13   The values 130 and 275 may need to be widened to
suit   the minimum and maximum flex levels being read by the   Analog pin */
 int flexReading = map(analogRead(flexPin), 130, 275, 4, 13);  //new command map, change 250, 900

// Make sure the value does not go beyond 4 or 13
 int LEDnum = constrain(flexReading, 4, 13); // new command constrain
```

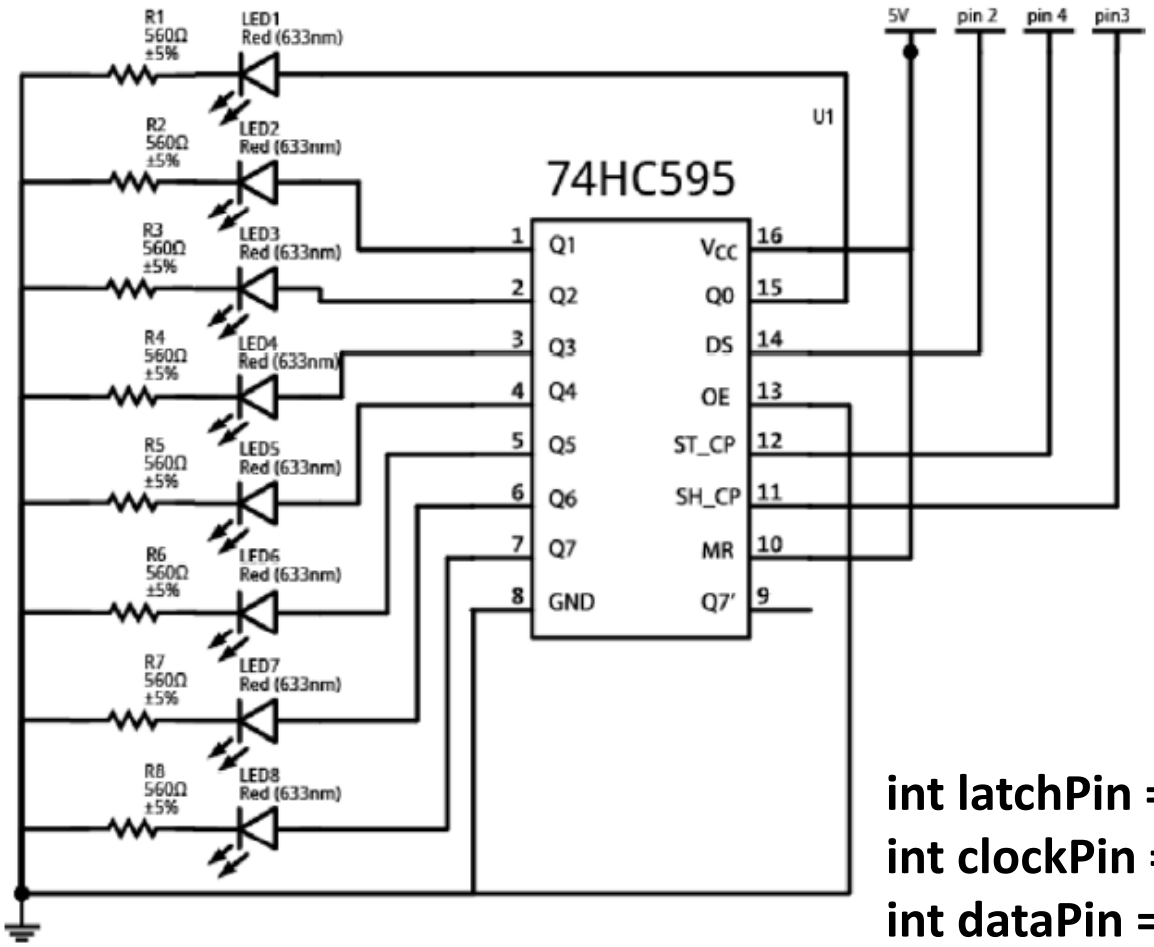
```
/*Call the blink function: this will turn the LED on for 10 milliseconds, and keep it
  off for only 1 millisecond. You can change the blink rate by changing these values,
  however, I want a quick response time when the flex sensor bends, hence the small
  values. LEDnum determines which LED gets turned on.*/
 blink(LEDnum, 10,1); //try changing to 100,100 for blinking
}

// The blink function - used to turn the LEDs on and off
void blink(int LEDPin, int onTime, int offTime){
  // Turn the LED on
 digitalWrite(LEDPin, HIGH);

  // Delay so that you can see the LED go On.
 delay(onTime);

  // Turn the LED Off
  digitalWrite(LEDPin, LOW);

 // Increase this Delay if you want to see an actual blinking effect.
  delay(offTime);
}
// END of Code
```

Shit Registers using the 74HC595

74HC595 IC:

- **Allows you to expand Arduino pins using 3 pins can get 8 outputs**
- **Great for driving lots of LEDs and expansion**
- **Uses synchronous serial communication**
- **Pulse one pin up and down communicating a data byte to the register bit by bit**
- **Second pin – clock separates them**
- **Third pin – sends the status of the bits out**






```
int latchPin = 4; //Pin connected to ST_CP of 74HC595
int clockPin = 3; //Pin connected to SH_CP of 74HC595
int dataPin = 2;////Pin connected to DS of 74HC595
```

The pieces of this diagram that are data related are:

- Q0 to Q7 (pin 1 to 7 and 15) are the eight pins containing the pattern. These pins are also know as the "output pins"
- Q7' (pin 9) is the bit that gets rolled (or shifted) out.
- DS (pin 14) is the bit that is getting rolled (shifted) in.

The pieces that are power and ground related are:

- GND (pin 8) is (you guessed it) connected to ground
- VCC (pin 16) is connected to 5V

The remaining four pins are used to control the pattern in the shift register:

- MR (pin 10) is the "**M**aster **R**eclear".
  - It will "empty" the whole shift register if pulled LOW. Notice that in our starting circuit, this pin is connected to 5V. This is because it must be pulled to HIGH to enable the shift register.
- SH_CP (pin 11) is the "**SH**ift register **C**lock **P**in".
  - When this pin is pulled HIGH, it will shift the register. This pin will alternate between HIGH and LOW. In our analogy, this is like the "bell" that indicates that a change will occur.
- ST_CP (pin 12) is the "**ST**orage register **C**lock **P**in".
  - Needs to be pulled to HIGH to have the "pattern" of bits be output by the shift register. This will be pulled HIGH after SH_CP has gone LOW. This pin will alternate between HIGH and LOW as well. This pin could be analogous to a "display" trigger. You could have multiple shifts, but only display the ending pattern.
- OE (pin 13) is the "**O**utput **E**nable" pin.
  - This pin enables the output when tied to GND and disables output when HIGH. Notice how our circuit has pin 13 tied to ground.

```
int latchPin = 4; //Pin connected to ST_CP of 74HC595
int clockPin = 3; //Pin connected to SH_CP of 74HC595
int dataPin = 2;////Pin connected to DS of 74HC595

void setup() {
  //set pins to output so you can control the shift register
  pinMode(latchPin, OUTPUT);
  pinMode(clockPin, OUTPUT);
  pinMode(dataPin, OUTPUT);
}
void loop() {
  // count from 0 to 255 and display the number
  // on the LEDs
  for (int numberToDisplay = 0; numberToDisplay < 256; numberToDisplay++) {
    // take the latchPin low so
    // the LEDs don't change while you're sending in bits:
    digitalWrite(latchPin, LOW);
    // shift out the bits:
    shiftOut(dataPin, clockPin, MSBFIRST, numberToDisplay);  // new function
    //take the latch pin high so the LEDs will light up:
    digitalWrite(latchPin, HIGH);
    // pause before next value:
    delay(500);
  } }
```

Another 74HC595 Sample Program

```
/* 74HC595 shift register attached to pins 2, 3, and 4 of the Arduino
 * LEDs attached to each of the outputs of the shift register     */

//Pin connected to latch pin (ST_CP) of 74HC595
const int latchPin = 4;
//Pin connected to clock pin (SH_CP) of 74HC595
const int clockPin = 3;
////Pin connected to Data in (DS) of 74HC595
const int dataPin = 2;

void setup() {
  //set pins to output because they are addressed in the main loop
  pinMode(latchPin, OUTPUT);
  pinMode(dataPin, OUTPUT);
  pinMode(clockPin, OUTPUT);
  Serial.begin(9600);
  Serial.println("reset");
}
void loop() {
  if (Serial.available() > 0) {
 // ASCII '0' through '9' characters are represented by the values 48 through 57, so if the user types a number from 0
through 9 in ASCII, you can subtract 48 to get the actual value:
    int bitToSet = Serial.read() - 48;
```

```
 // write to the shift register with the correct bit set high:
   registerWrite(bitToSet, HIGH);
  }
}

// This method sends bits to the shift register:

void registerWrite(int whichPin, int whichState) {
// the bits you want to send
  byte bitsToSend = 0;

  // turn off the output so the pins don't light up
  // while you're shifting bits:
  digitalWrite(latchPin, LOW);

  // turn on the next highest bit in bitsToSend:
  bitWrite(bitsToSend, whichPin, whichState);

  // shift the bits out:
  shiftOut(dataPin, clockPin, MSBFIRST, bitsToSend);
// LED On
  digitalWrite(latchPin, HIGH);
}
```

Switch and Case , While

Provide a menu and based on user input perform a function or run a command.

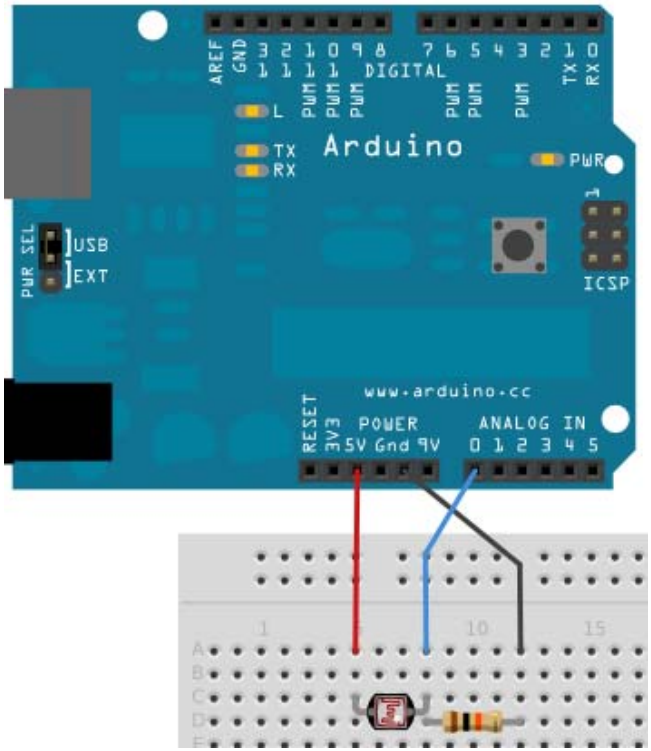
Example 1: Using a CDS depending on value select from a list of cases

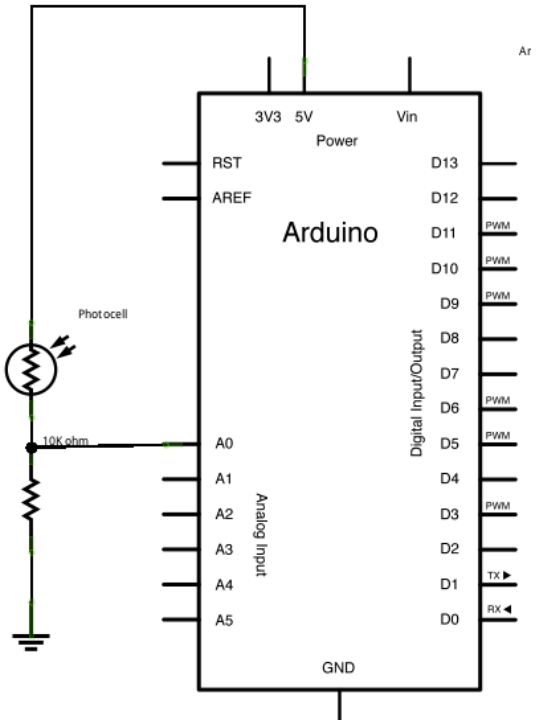
Example 2: Magic ball, activate a switch and the program randomly selects an option from the case statements to provide you with an answer

Example 3: Type a letter on the Serial Monitor and it turns on an LED based on the letter you press.

Example 4: While button is pressed check sensor values to see if it is calibrated

Example 1: The photoresistor is connected to analog in pin 0 using a voltage divider circuit. A 10Kilohm resistor makes up the other side of the voltage divider, running from analog in 0 to ground. The analogRead() function returns a range of about 0 to 600 from this circuit in a reasonably lit indoor space.





```
/*   Switch statement - Demonstrates the use of a switch statement.  The switch statement allows you to choose from among
a set of discrete values of a variable.  It's like a series of if statements.  To see this sketch in action, but the board and sensor in
a well-lit room, open the serial monitor, and move your hand gradually down over the sensor.
  The circuit:  * photoresistor from analog in 0 to +5V,   * 10K resistor from analog in 0 to ground
  created 1 Jul 2009  modified 9 Apr 2012  by Tom Igoe
    This example code is in the public domain.  http://www.arduino.cc/en/Tutorial/SwitchCase  */

// these constants won't change. They are the
// lowest and highest readings you get from your sensor:
 const int sensorMin = 0;      // sensor minimum, discovered through experiment
 const int sensorMax = 600;    // sensor maximum, discovered through experiment

void setup() {
   // initialize serial communication:
   Serial.begin(9600);
}
void loop() {
   // read the sensor:
   int sensorReading = analogRead(A0);
   // map the sensor range to a range of four options:
   int range = map(sensorReading, sensorMin, sensorMax, 0, 3);

   // do something different depending on the range value:
```

```
 switch (range) {
   case 0:    // your hand is on the sensor
    Serial.println("dark");
     break;
   case 1:    // your hand is close to the sensor
     Serial.println("dim");
     break;
   case 2:    // your hand is a few inches from the sensor
     Serial.println("medium");
     break;
   case 3:    // your hand is nowhere near the sensor
     Serial.println("bright");
     break;
   }
   delay(1);        // delay in between reads for stability
}

// try left, middle, right with potentiometer
```

Example 2: Magicball, activate a switch and the program randomly selects an option from the case statements to provide you with an answer

```
// EXAMPLE 2:  Crystal Ball
#include <LiquidCrystal.h>  // include the library code:
// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
// set up a constant for the tilt switchPin
const int switchPin = 6;   // variable to hold the value of the switchPin
int switchState = 0;         // variable to hold previous value of the switchpin
int prevSwitchState = 0;  // a variable to choose which reply from the crystal ball
int reply;

void setup() {
  // set up the number of columns and rows on the LCD
  lcd.begin(16, 2);
    // set up the switch pin as an input
  pinMode(switchPin,INPUT);
  // Print a message to the LCD.
  lcd.print("Ask the");
  // set the cursor to column 0, line 1
  // line 1 is the second row, since counting begins with 0
  lcd.setCursor(0, 1);
  // print to the second line
  lcd.print("Crystal Ball!");
}
```

```
void loop() {
  // check the status of the switch
  switchState = digitalRead(switchPin);

  // compare the switchState to its previous state
  if (switchState != prevSwitchState)
{
    // if the state has changed from HIGH to LOW
    // you know that the ball has been tilted from  one direction to the other

    if (switchState == LOW) {
      // randomly chose a reply
      reply = random(8);
      // clean up the screen before printing a new reply
      lcd.clear();

      // set the cursor to column 0, line 0
      lcd.setCursor(0, 0);

      // print some text
      lcd.print("the ball says:");

      // move the cursor to the second line
      lcd.setCursor(0, 1);
```

```
 // choose a saying to print based on the value in reply
     switch(reply){
     case 0:
       lcd.print("Yes");
       break;
     case 1:
       lcd.print("Most likely");
       break;
     case 2:
       lcd.print("Certainly");
       break;
     case 3:
       lcd.print("Outlook good");
       break;
     case 4:
       lcd.print("Unsure");
       break;
     case 5:
       lcd.print("Ask again");
       break;
     case 6:
       lcd.print("Doubtful");
       break;
     case 7:
       lcd.print("No");
       break;
     }   } }
  // save the current switch state as the last state
  prevSwitchState = switchState;
} // roulette game anyone?
```

Example 3: Type a letter on the Serial Monitor and it turns on an LED based on the letter you press.

// **Example 3:**

```
/*   Switch statement  with serial input.  Demonstrates the use of a switch statement.  The switch statement allows you to choose from among a set of discrete values of a variable.  It's like a series of if statements.  To see this sketch in action, open the Serial monitor and send any character.  The characters a, b, c, d, and e, will turn on LEDs.  Any other character will turn the LEDs off.
 I've modified it to turn on built-in LED on pin13 when letter a is input 12-22-14 - walter
 The circuit: 5 LEDs attached to digital pins 2 through 6 through 220-ohm resistors */

void setup() {
  // initialize serial communication:
  Serial.begin(9600);
   // initialize the LED pins:
      for (int thisPin = 2; thisPin < 14; thisPin++) {
        pinMode(thisPin, OUTPUT);
      }   }
void loop() {
  // read the sensor:
  if (Serial.available() > 0) {
    int inByte = Serial.read();
/* do something different depending on the character received.  The switch statement expects single number values for each case; in this exmaple, though, you're using single quotes to tell the controller to get the ASCII value for the character. For  example 'a' = 97, 'b' = 98, and so forth: */

digitalWrite(13, LOW);  // I set the pin to LOW initially
```

```
switch (inByte) {
    case 'a':
      digitalWrite(13, HIGH);
      delay (1500);
      break;
    case 'b':
      digitalWrite(8, HIGH);
delay (1500);
      break;
    case 'c':
      digitalWrite(4, HIGH);
      break;
    case 'd':
      digitalWrite(5, HIGH);
      break;
    case 'e':
      digitalWrite(6, HIGH);
      break;
    default:
      // turn all the LEDs off:
      for (int thisPin = 2; thisPin < 7; thisPin++) {
        digitalWrite(thisPin, LOW);
      }   }   }   }
```

Example 4: Conditional While statement. While you press a switch start a calibration process

```
/*  Conditionals - while statement
  This example demonstrates the use of  while() statements.  While the pushbutton is pressed, the sketch runs the
calibration routine. The  sensor readings during the while loop define the minimum and maximum
 of expected values from the photo resistor. The circuit: * photo resistor connected from +5V to analog in pin 0
 * 10K resistor connected from ground to analog in pin 0
 * LED connected from digital pin 9 to ground through 220 ohm resistor
 * pushbutton attached from pin 2 to +5V
 * 10K resistor attached from pin 2 to ground  */
// These constants won't change:
const int sensorPin = A0;       // pin that the sensor is attached to
const int ledPin = 8;           // pin that the LED is attached to
const int indicatorLedPin =13; // pin that the built-in LED is attached to
const int buttonPin = 2;        // pin that the button is attached to
// These variables will change:
int sensorMin = 1023;  // minimum sensor value
int sensorMax = 0;     // maximum sensor value
int sensorValue = 0;         // the sensor value
void setup() {
  // set the LED pins as outputs and the switch pin as input:
  pinMode(indicatorLedPin, OUTPUT);
  pinMode (ledPin, OUTPUT);
  pinMode (buttonPin, INPUT);   }
```

```
void loop() {
  // while the button is pressed, take calibration readings:
  while (digitalRead(buttonPin) == HIGH) {
    calibrate();
  }
  // signal the end of the calibration period
  digitalWrite(indicatorLedPin, LOW);
  // read the sensor:
  sensorValue = analogRead(sensorPin);
  // apply the calibration to the sensor reading
  sensorValue = map(sensorValue, sensorMin, sensorMax, 0, 255);
  // in case the sensor value is outside the range seen during calibration
  sensorValue = constrain(sensorValue, 0, 255);
  // fade the LED using the calibrated value:
  analogWrite(ledPin, sensorValue);
}
```

```
void calibrate() {
  // turn on the indicator LED to indicate that calibration is happening:
  digitalWrite(indicatorLedPin, HIGH);
  // read the sensor:
  sensorValue = analogRead(sensorPin);
  // record the maximum sensor value
  if (sensorValue > sensorMax) {
    sensorMax = sensorValue;
  }
  // record the minimum sensor value
  if (sensorValue < sensorMin) {
    sensorMin = sensorValue;
  }   }
```

Note: Basically, the WHILE loop will loop until it reads a LOW from the buttonPin pin port number. In other words – when the button sends a LOW – or not pressed – then the WHILE loop will stop looping.

Example to check the state of the button (how many times has it been pressed?):

```
/* State change detection (edge detection)
Often, you don't need to know the state of a digital input all the time, but you just need to know when the input changes from one state to another. For example, you want to know when a button goes from OFF to ON.  This is called
state change detection, or edge detection.  This example shows how to detect when a button or button changes from off to on and on to off.
  The circuit:
  * pushbutton attached to pin 2 from +5V
  * 10K resistor attached to pin 2 from ground
  * LED attached from pin 13 to ground (or use the built-in LED on most Arduino boards)
http://arduino.cc/en/Tutorial/ButtonStateChange */

// this constant won't change:
const int  buttonPin = 2;    // the pin that the pushbutton is attached to

const int ledPin = 13;       // the pin that the LED is attached to

// Variables will change:

int buttonPushCounter = 0;   // counter for the number of button presses

int buttonState = 0;         // current state of the button

int lastButtonState = 0;     // previous state of the button
```

```
void setup() {
pinMode(buttonPin, INPUT);
pinMode(ledPin, OUTPUT);
  Serial.begin(9600);
 }

void loop() {
   // read the pushbutton input pin:
   buttonState = digitalRead(buttonPin);
   // compare the buttonState to its previous state
    if (buttonState != lastButtonState) {
     // if the state has changed, increment the counter
     if (buttonState == HIGH) {
       // if the current state is HIGH then the button
      buttonPushCounter++;
       Serial.println("on");
       Serial.print("number of button pushes:  ");
       Serial.println(buttonPushCounter);
     }
```

```
   else {
      // if the current state is LOW then the button
Serial.println("off");
    }   }
  // save the current state as the last state for next time through the loop
  lastButtonState = buttonState;
  // turns on the LED every four button pushes by checking the modulo of the button push counter.
  // the modulo function gives you the remainder of the division of two numbers:
  if (buttonPushCounter % 4 == 0) {
    digitalWrite(ledPin, HIGH);
  }
 else {
    digitalWrite(ledPin, LOW);
  }
 }
```

Reference:

https://opensourcehardwaregroup.com/tutorial-18-**state-change-detection-and-the-modulo-operator**-old-version/

Take a look at the following calculations and see if you can follow along:

5 % 2 = 1

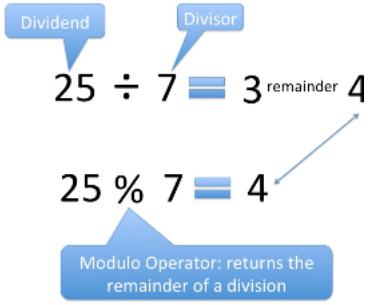
17 % 5 = 2

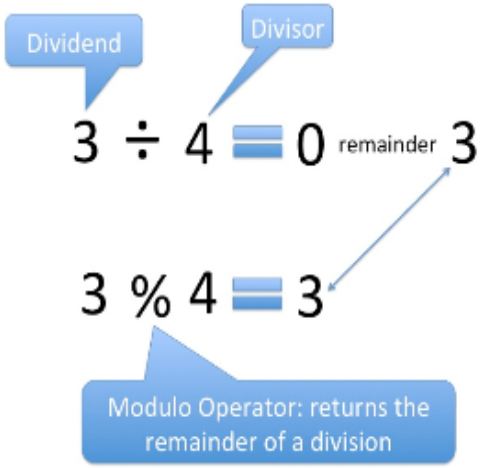
1 % 4 = 1

2 % 4 = 2

3 % 4 = 3

4 % 4 = 0





MODULO OPERATOR

if (buttonPushCounter % 4 == 0)
Where buttonPushCounter = 4
So 4/4 has a remainder of 0

If we are looking for 3 buton presses the we would use 3%4 because 4 does not go into 3
So the remainder is 3

So if you divided 2 into 5, what would be the remainder?

5/2 = 2 remainder 1

So 2 goes into five twice with one left over.

Sensors and Devices Preview–

- **Sensors can be both binary or a range.**
- **Usually, sensors that measure a range of values vary their resistance to reflect their detection. (analog)**
- **Arduinos can only sense voltages, not resistances.**
- **Sensors that only vary their resistances require a circuit called a voltage divider to provide the Arduino a voltage.**
- **Input Sensors: switch (digital), variable resistor (analog)**
- **Output Devices: speaker, LED, Motor**