PROGRAMS AND EXERCISES

Dynamic Educational Services – level 2 courseware

www.dynamicedu.in/botdulars/

THE HELLO WORLD PROGRAM FOR ROBOTICS !

- Program 1 – is a simple set of instructions to turn the LED bulbs ON and OFF
- The LED bulbs are connected to digital pins 7, 8 and 3
- The setup section initializes them as OUTPUT pins and then the loop section turns them ON and OFF on a sequence.

Simple exercises

1. Turn the LED bulbs ON and OFF in a different sequence
2. Turn Red ON, 1 second after Green and turn OFF green 2 seconds later
3. Understand difference between instructions in Setup block vs. the Loop

```
void changeLights(){
    // green off, yellow on for 3 seconds
    digitalWrite(green, LOW);
    digitalWrite(yellow, HIGH);
    delay(3000);

    // turn off yellow, then turn red on for 3 seconds
    digitalWrite(yellow, LOW);
    digitalWrite(red, HIGH);
    delay(3000);

    // red and yellow on for 2 seconds (red is already on though)
    digitalWrite(yellow, HIGH);
    delay(2000);

    // turn off red and yellow, then turn on green
    digitalWrite(yellow, LOW);
    digitalWrite(red, LOW);
    digitalWrite(green, HIGH);
    delay(3000);
}
```

PROGRAM 2- UNDERSTANDING DIGITAL INPUT

- Program 2 – takes instruction from a PUSH button and turns ON the LED
- Digital pin for Input (push button) is 12 and the LED is at 7

Simple exercises

1. Try the other LED as well to be turned ON along with this
2. Toggle between RED and GREEN LED
3. Print count of button presses on the screen

```
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 == LOW) {
    // turn LED on:
    digitalWrite(ledPin, HIGH);
  }
  else {
    // turn LED off:
    digitalWrite(ledPin, LOW);
  }
}
```

PROGRAM 3- ACTIVATE BUZZER UPON INPUT

- Program 3 – is similar to program 2 but activates the buzzer instead of the LED
- Digital pin for Input (push button) is 7 and the Buzzer is at 4

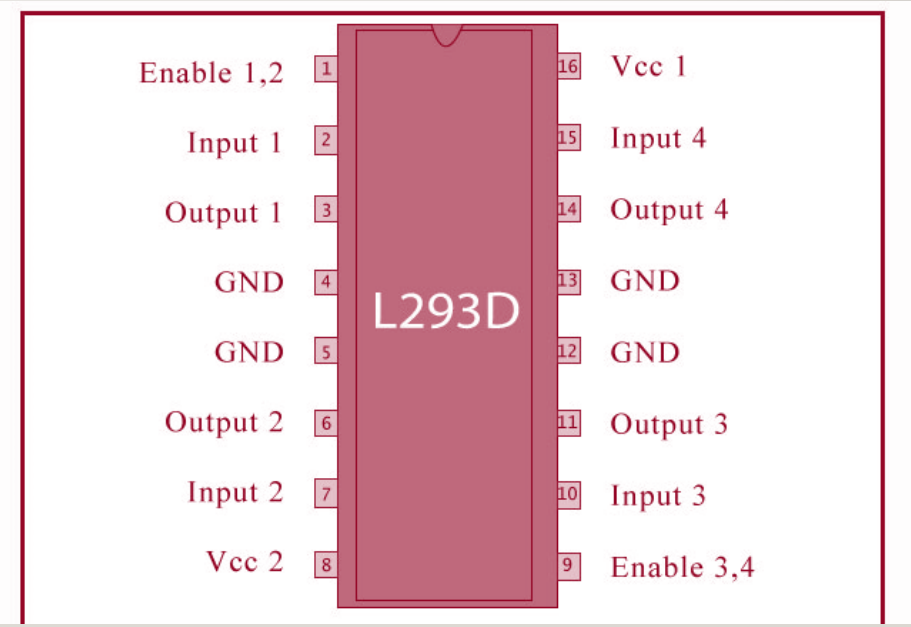
Simple exercises

1. Check the significance of button state being 0 or 1
2. Increase the seconds for which the buzzer is activated
3. Do the same exercise using a breadboard

```
void loop(){
  buttonState = digitalRead(buttonPin);
  Serial.println(buttonState);
  digitalWrite(buzzer, HIGH);
  noTone(buzzer);
   if (buttonState == LOW){
      digitalWrite(buzzer, LOW);
      //tone(buzzer, 1000); // Send 1KHz sound signal...
      delay(1000);        // ...for 1 sec
   }
   else
   {
      digitalWrite(buzzer, HIGH) ;
      delay(1000);        // ...for 1sec
   }

}
```

CONCEPTS, FOR PROGRAM 4- WHAT IS AN L298 OR L293 DRIVER?

- Control speed and direction
- Takes the PWM signal from Arduino over to the Motors
- Concept of a H-bridge to reverse



PROGRAM 4- H-BRIDGE TO MOVE MOTORS IN BOTH DIRECTIONS

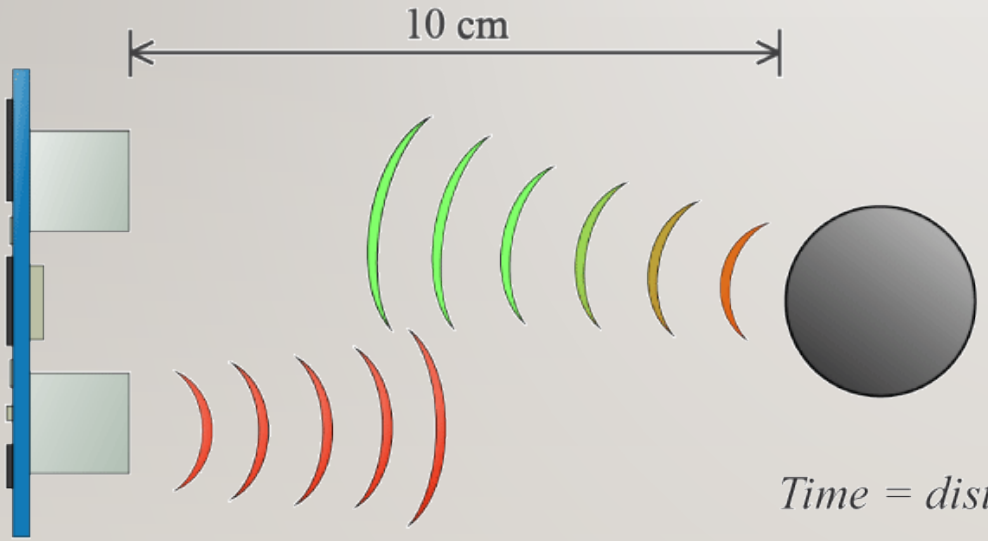
- L293D is the H-bridge used to reverse polarity and achieve bi-directional movement
- Notice that there is no code within the loop section and we are still able to test the working of the motors

Simple exercises

1. Rotate Motor 2 before Motor 1 and play with the seconds
2. Try AnalogWrite functions instead of DigitalWrite
3. Activate motor on pressing of the button

```
//botdulars sample code to test L293D connection with the below pins
//L293D
//Motor A
const int motorPin1  = 10;  // Pin 14 of L293
const int motorPin2  = 9;  // Pin 10 of L293
//Motor B
const int motorPin3  = 6; // Pin  7 of L293
const int motorPin4  = 5;  // Pin  2 of L293

//This will run only one time.
void setup(){

   //Set pins as outputs
   pinMode(motorPin1, OUTPUT);    pinMode(motorPin2, OUTPUT);  pinMode(motorPin3, OUTPUT);
pinMode(motorPin4, OUTPUT);
   //This code  will turn Motor A clockwise for 1/2 sec.
   digitalWrite(motorPin1, HIGH);  digitalWrite(motorPin2, LOW);
   digitalWrite(motorPin3, LOW);   digitalWrite(motorPin4, LOW);
   delay(500);
   //This code will turn Motor A counter-clockwise for 1/2 sec.
   digitalWrite(motorPin1, LOW);     digitalWrite(motorPin2, HIGH);
   digitalWrite(motorPin3, LOW);     digitalWrite(motorPin4, LOW);
   delay(500);
   Serial.begin(9600);
   //This code will turn Motor B clockwise for 1/2 sec.
   digitalWrite(motorPin1, LOW);   digitalWrite(motorPin2, LOW);
   digitalWrite(motorPin3, HIGH);  digitalWrite(motorPin4, LOW);
   delay(500);
   //This code will turn Motor B counter-clockwise for 1/2 sec.
   digitalWrite(motorPin1, LOW);   digitalWrite(motorPin2, LOW);
   digitalWrite(motorPin3, LOW);   digitalWrite(motorPin4, HIGH);
   delay(500);
   Serial.println("clockwise and counter clockwise test for H-bridge done. Did the wheels spin both ways ?");
   //And this code will stop motors
   digitalWrite(motorPin1, LOW);    digitalWrite(motorPin2, LOW);
   digitalWrite(motorPin3, LOW);    digitalWrite(motorPin4, LOW);
}

void loop(){
}
```

CONCEPT FOR PROGRAM 5 – WHAT IS THE SR-04 ULTRASONIC SENSOR?



speed of sound:

$v = 340\ m/s$

$v = 0{,}034\ cm/\mu s$

Time = distance / speed:

$t = s / v = 10 / 0{,}034 = 294\ \mu s$

Distance:

$s = t \cdot 0{,}034 / 2$

PROGRAM 5- ULTRA-SONIC SENSOR TO CALCULATE DISTANCE OF OBSTACLE

- SR-04 is the sensor used to calculate this distance. Notice that this is connected to the Analog pins and one being OUTPUT while the other is INPUT.
- The approach to calculation of distance is similar to how radars or even bats operate, by bouncing signal off the nearest obstacle

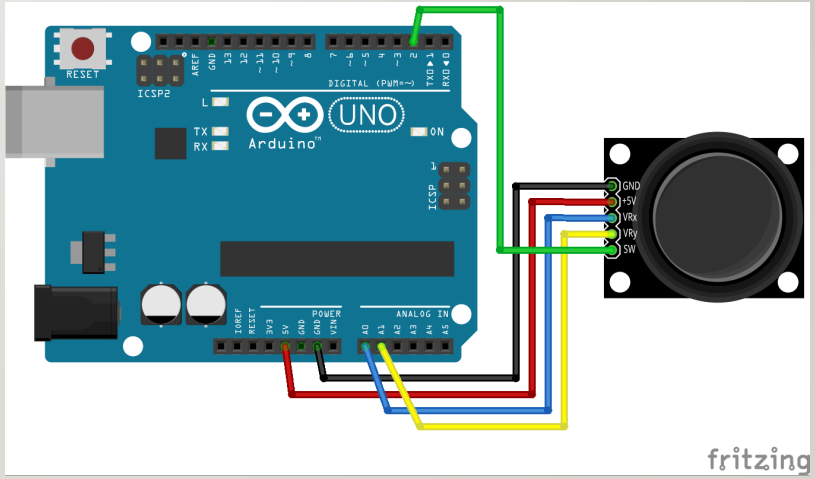
Simple exercises

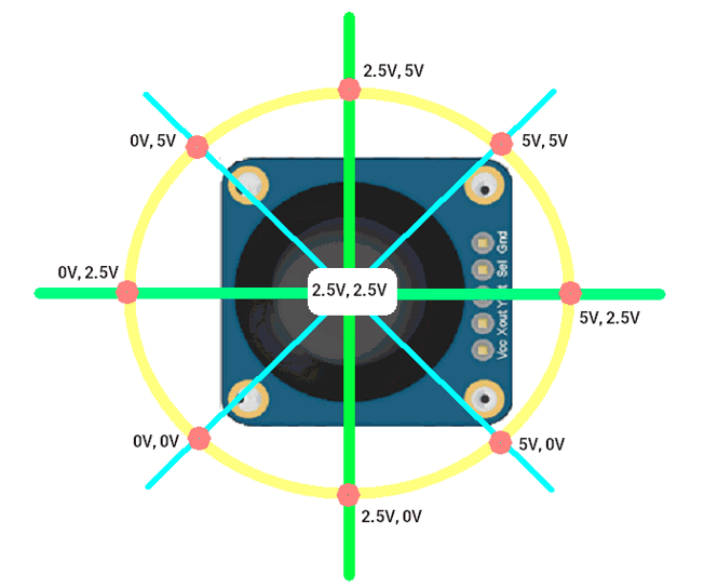
1. Vary the obstacle in front and validate distance measurements
2. Turn ON the LED if the distance is within a range
3. Write 2 examples of where Obstacle sensor is used in real world

```
const int trigPin = A3;
const int echoPin = A4;

float duration, distance;

void setup() {
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
  Serial.begin(9600);
}

void loop() {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  duration = pulseIn(echoPin, HIGH);
  distance = (duration*.0343)/2;
  Serial.print("Distance: ");
  Serial.println(distance);
  delay(2000);
}
```

CONCEPTS INVOLVED – UNDERSTANDING JOYSTICK WIRING TO ARDUINO

- The joystick can move across the X and Y axes
- Two potentiometers do the job for us in providing various values as we turn the Joystick around





- In Electronics terminology, this behaves as a Voltage divider network
- At this point, suffice it to say that a Voltage divider network has resistors connected within itself in series, in order that the output voltage is regulated in varying magnitudes depending on the position of the potentiometer

PROGRAM 6- JOYSTICK AS AN INPUT DEVICE

- Understand Analog inputs from the Joystick device attached to the respective Pins
- Understand how the mapping of inputs from the device is translated to a range using the map function

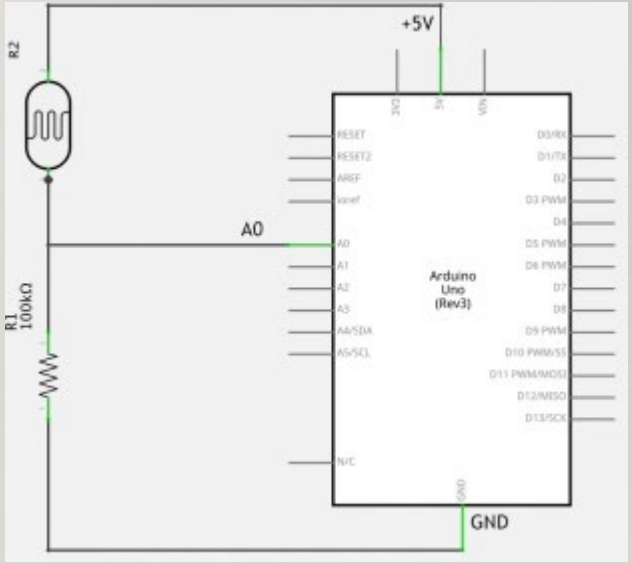
Simple exercises

1. Vary the values within the map function to see the output change
2. Upon button state being zero (clicked) can you turn ON an LED?
3. Check on the link below to convert the output to actual voltage instead of using the map function.

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

  pinMode(VRx, INPUT);
  pinMode(VRy, INPUT);
  pinMode(A2, INPUT);
  digitalWrite(A2, HIGH);
  pinMode(A2, INPUT_PULLUP);

}

void loop() {
  xPosition = analogRead(VRx);
  yPosition = analogRead(VRy);
  SW_state = digitalRead(SW);
  mapX = map(xPosition, 0, 1023, -512, 512);
  mapY = map(yPosition, 0, 1023, -512, 512);

  Serial.print("X: ");
  Serial.print(mapX);
  Serial.print(" | Y: ");
  Serial.print(mapY);
  Serial.print(" | Button: ");
  Serial.println(SW_state);

  delay(1000);

}
```

CONCEPTS INVOLVED – UNDERSTANDING LDR

- LDR stands for being a Light Dependent Resistor where the resistance becomes less when the light is more and vice versa
- This is another sensor that is best connected to the Analog pins (given it gives us varying values as input and not just Zero or One)
- The schematic is simple and given alongside.



PROGRAM 7- LIGHT DEPENDENT RESISTOR

- Understand varying output values based on light around the LDR.
- A simple conditional program is represented alongside to turn ON the LED when there is lesser light on the LDR

Simple exercises

1. Use cloth or hands to vary ambient light
2. Can you think of 2 real world requirements for LDR ?
3. What would happen if we connected LDR to a digital pin?

```
void loop() {
  int ldrStatus = analogRead(ldrPin);
  if (ldrStatus <= 200) {
    digitalWrite(ledPin, HIGH);
    Serial.print("Its DARK, Turn on the LED : ");
    Serial.println(ldrStatus);
  } else {
    digitalWrite(ledPin, LOW);
    Serial.print("Its BRIGHT, Turn off the LED : ");
    Serial.println(ldrStatus);
  }
}
```

PROGRAM 8- POTENTIOMETER AS AN INPUT DEVICE

- Potentiometer converts rotary motion to a change of resistance that is then read by the Analog pins
- A simple conditional program is represented alongside to turn ON the LED when the output voltage is sufficient enough from the PWM pins

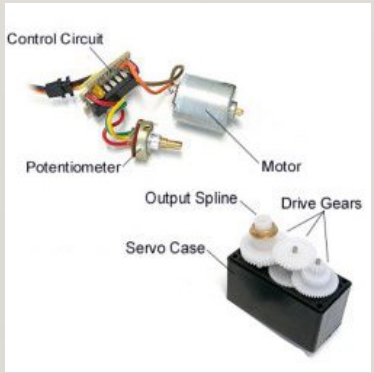
Simple exercises

1. Based on the concept that 5v is the max output, can you do some simple math to determine minimum voltage required (from the println output) to turn ON the LED ?
2. Refresh – what are PWM pins ? Why did we have to map the output to range between 0 and 255?

```
//botdulars sample code for potentiometer

void setup(){
  //Input or output?
  pinMode(ledPin, OUTPUT);
  pinMode(potPin, INPUT); //Optional
  Serial.begin(9600) ;
}

void loop(){

  value = analogRead(potPin);          //Read and save analog value from
potentiometer
  value = map(value, 0, 1023, 0, 255); //Map value 0-1023 to 0-255 (PWM)
  analogWrite(ledPin, value);          //Send PWM value to led
  Serial.println(value) ;
  delay(1000);                         //Small delay

}
```

CONCEPT INVOLVED- UNDERSTANDING SERVO

- Servo motors are seen in toy cars, robots for fixed and efficient movements
- Inside, you will notice that a DC motor is controlled by a gear mechanism that limits the range of movement based on the input received
- You will also notice that a sweep ∠ of 90 to 180 degrees is common across these servos



PROGRAM 9- MIMIC ARM MOVEMENT USING SERVO MOTORS

- The Arduino IDE itself comes with Sweep and Knob type examples
- A simple program to demonstrate the 180 degree sweep ∠ of the Servo is captured here

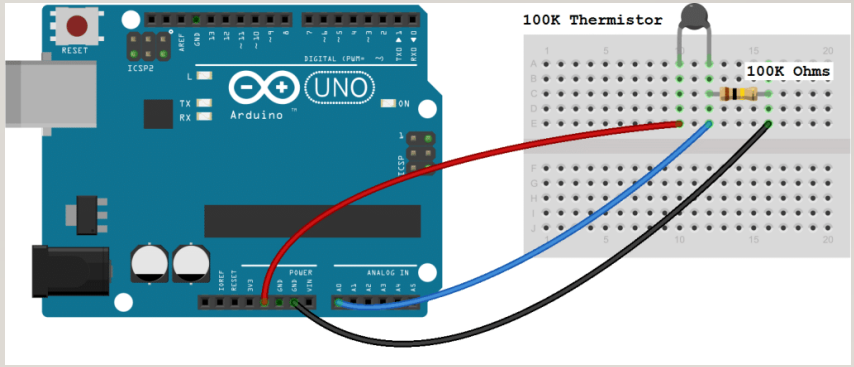
Simple exercises

1. Try out the sample programs given with Arduino IDE
2. Refresh – significance of PWM pins for operating motors like the Servo

```
void setup() {
  myservo.attach(11);
  for (pos = 0; pos <= 180; pos += 5) { // goes from 0 degrees to 180
    myservo.write(pos);              // tell servo to go to position
    delay(15);                       // waits 15ms for the servo to reach position
  }
  for (pos = 180; pos >= 0; pos -= 5) { // goes from 180 degrees to 0
    myservo.write(pos);              // tell servo to go to position in variable
    delay(15);                       // waits 15ms for the servo to reach
  }

}

void loop() {
}
```

CONCEPT INVOLVED – UNDERSTANDING THERMISTORS

- Thermistors are cost effective temperature measuring components and more importantly, smooth, simple Analog based sensors
- The thermistors are made of material (such as metal oxide or ceramic) that ensure correlation between resistance output and temperature. There are two types of thermistors – one, where the relationship is directly proportional and two- where the relationship is inversely proportional
- What we have in our kit is called NTC – Negative Temperature Coefficient thermistors where resistance decreases with increase in temperature



PROGRAM 10- THERMISTOR FOR MEASURING TEMPERATURE AND RESISTANCE

- The Steinhart-Hart equation helps us calculate the resistance and temperature in this example

Simple exercises

1. Try covering the sensor in your hand to see a change in temperature/ resistance
2. Do you remember difference between log to the base e and log to the base 10 ?

```
void loop() {
  int thermistor_adc_val;
  double output_voltage, thermistor_resistance, therm_res_ln, temperature;
  thermistor_adc_val = analogRead(thermistor_output);
  output_voltage = ( (thermistor_adc_val * 5.0) / 1023.0 );
  thermistor_resistance = ( ( 5 * ( 10.0 / output_voltage ) ) - 10 ); /* Resistance
in kilo ohms */
  thermistor_resistance = thermistor_resistance * 1000 ; /* Resistance in ohms
*/
  therm_res_ln = log(thermistor_resistance);
  /*  Steinhart-Hart Thermistor Equation: */
  /*  Temperature in Kelvin = 1 / (A + B[ln(R)] + C[ln(R)]^3)   */
  /*  where A = 0.001129148, B = 0.000234125 and C = 8.76741*10^-8  */
  temperature = ( 1 / ( 0.001129148 + ( 0.000234125 * therm_res_ln ) + (
0.0000000876741 * therm_res_ln * therm_res_ln * therm_res_ln ) ) ); /*
Temperature in Kelvin */
  temperature = temperature - 273.15; /* Temperature in degree Celsius */
  Serial.print("Temperature in degree Celsius = ");
  Serial.print(temperature);
  Serial.print("\t\t");
  Serial.print("Resistance in ohms = ");
  Serial.print(thermistor_resistance);
  Serial.print("\n\n");
  delay(2000);
}
```

CONCEPT INVOLVED

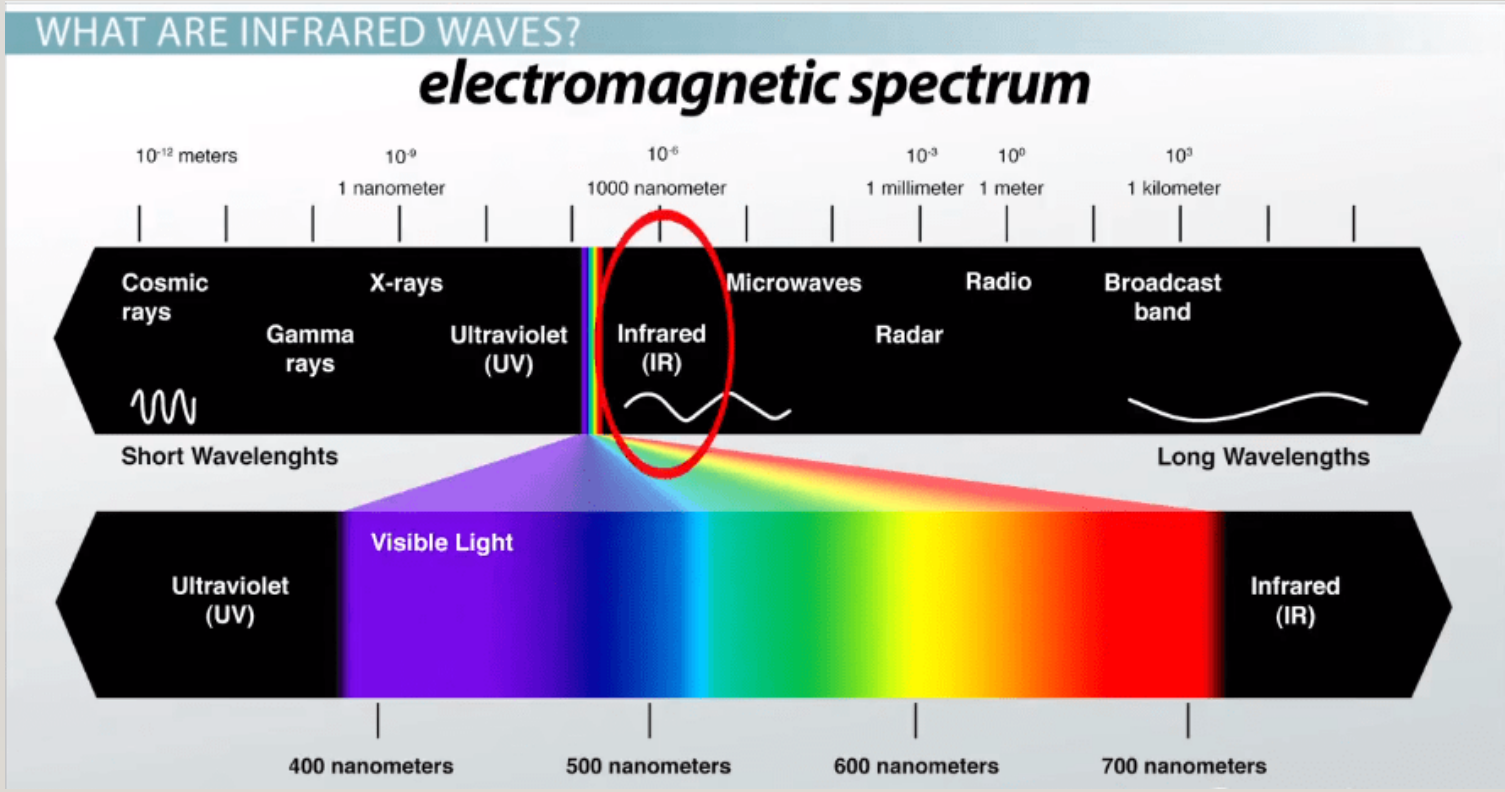
- What is an Infra red signal
  - What is a light spectrum
  - IR uses a frequency above the range of visible light
  - Predominantly, uses the line-of-sight mode
- What are the libraries and functions used

IRrecv irrecv(receivePin)Create the receiver object, using a name of your choice

irrecv.enableIRIn()Begin the receiving process. This will enable the timer interrupt which consumes a small amount of CPU every 50 µs.

irrecv.decode(&results)Attempt to receive a IR code. Returns true if a code was received, or false if nothing received yet. When a code is received, information is stored into "results".

CONCEPTS, CONTINUED …



PROGRAM 11- RECOGNIZE AND RECEIVE VALUES FROM IR REMOTE

- Note how the decoded values differ for each button
- There is a specific call to delay within the loop to make sure sufficient time is given, before we resume polling for the next

Simple exercises

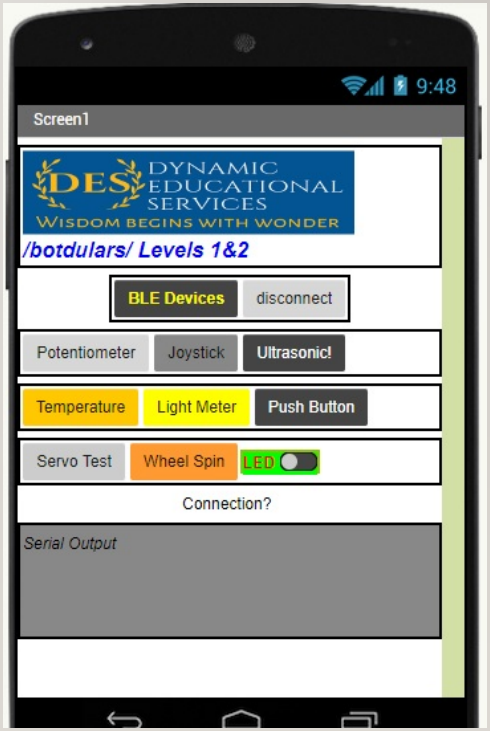
1. Can you turn ON an LED upon a specific button press ?
2. Can you try a home remote and see the kind of Hex values that come up for each button press?

```
//botdulars sample – for IR receive signals

IRrecv irrecv(RECV_PIN);

decode_results results;

void setup()
{
  Serial.begin(9600);
  // In case the interrupt driver crashes on setup, give a clue
  // to the user what's going on.
  Serial.println("Enabling IRin");
  irrecv.enableIRIn(); // Start the receiver
  Serial.println("Enabled IRin");
}

void loop() {
  if (irrecv.decode(&results)) {
    Serial.println(results.value, HEX);
    delay(250);
    //dump(&results);
    irrecv.resume(); // Receive the next value
  }
}
```

WHAT ARE THE FINAL PROJECT CHOICES?

- A IR remote controlled car
- A Bluetooth app controlled car
- A Bluetooth app that shows LIVE readings of temperature, brightness, joystick and potentiometer readings and output from the distance sensor
- A home alarm system that beeps and lights up when objects come too close

www.github.com/botdulars/apk has a sample apk created using MIT app inventor that can be used



THANK YOU

www.dynamicedu.in

meera@dynamicedu.in