EXPERIMENT 7

Please visit https://www.arduino.cc/en/Reference/HomePage

to learn all features of arduino before you start the experiments

TEMPERATURE MEASUREMENT AND CONTROL

USING LM35

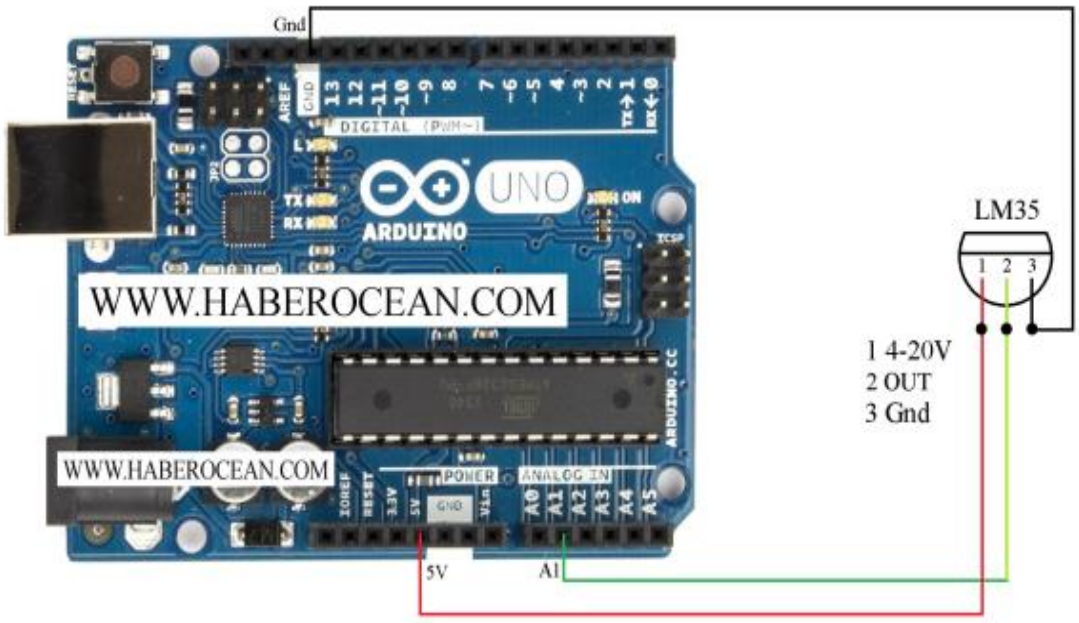
Purpose: To measure and control the room temperature using LM35 and Arduino UNO.

Description:

LM35 is a precision temperature integrated circuit which gives an output voltage proportional to the temperature in centigrade. LM35 has three legs out of which two are positive voltage supply and ground. Remaining one is the output which gives an output voltage proportional to temperature in centigrade. Circuit is done as shown in the following diagram.

Connecting LM35 to Arduino UNO:

Connect the Vcc, GND and OUT pins of the LM35 to the Arduino pins as follows:



Connect LED1 to the pin 9 and a LED2 to pin 8 via a series resistance of 330 ohm. Also connect a relay to pin 7 to control the operation of a heater.

A program will be written to control the room temperature according to the following specifications:

The heater will be turned ON when the Temperature > 27 deg C and LED1 on pin 9 will be ON to indicate that the heater is ON. LED2 will be OFF in this case.

The heater will be turned OFF when the Temperature < 26 deg C and LED2 on pin 8 will be ON to indicate that the heater is OFF while LED1 will be turned OFF.

Write the following program on the Arduino sketch and download it to the Arduino UNO

```
/* This program demonstrates the use of LM35 temperature sensor.
   Analog output of LM35 is connected to A0. A relay is connected
   to pin 7 to control a heater. LED1 on pin 9 indicates that the
   heater is ON and LED2 on pin 8 indicates that the heater is OFF.
   The heater is turned on when the Temp<26 and is is turned OFF when
   Temp>27. A relay on pin 7 is turned ON and OFF to control a heater with
   temperature changes.
 */
void setup()
{
  pinMode(7, OUTPUT); //Relay is connected to pin7
  pinMode(8, OUTPUT); //Heater OFF indicator (LED2)
  pinMode(9, OUTPUT); //Heater ON indicator (LED1)
  Serial.begin(9600);
  Serial.print("TEMPERATURE");
 }
void loop()
{
 int analogValue = analogRead(A0);   // Read the raw temperature value from LM35 sensor
 float Temp = analogValue*500/1023; //Convert the raw value to real temperature (10 mV/deg)
if (Temp>27) {
 digitalWrite(7, LOW); //Relay on pin 7 is turned OFF to turn the heater OFF
  digitalWrite(9, LOW); // LED1 indicator on pin 9 is turned OFF
 digitalWrite(8, HIGH); // LED2 indicator on pin 8 is turned ON
 }
 if (Temp<26){
 digitalWrite(7, HIGH); //Relay on pin 7 is turned ON to turn the heater ON
 digitalWrite(9, HIGH); // LED2 indicator on pin 9 is turned ON
 digitalWrite(8, LOW); // LED1 indicator on pin 8 is turned OFF
  }
   Serial.print("TEMPERATURE =  ");
   Serial.println(Temp);
 delay(100) ; // wait for 100 ms
}
```

Serial Communication via UART

Used for communication between the Arduino board and a computer or other devices. All Arduino boards have at least one serial port (also known as a UART or USART): Serial. It communicates on digital pins 0 (RX) and 1 (TX) as well as with the computer via USB. Thus, if you use these functions, you cannot also use pins 0 and 1 for digital input or output.

You can use the Arduino environment's built-in serial monitor to communicate with an Arduino board. Click the serial monitor button in the toolbar and select the same baud rate used in the call to begin().

available()

Description

Get the number of bytes (characters) available for reading from the serial port. This is data that's already arrived and stored in the serial receive buffer (which holds 64 bytes). available() inherits from the Stream utility class.

Syntax

Serial.available()

```
int incomingByte = 0;   // for incoming serial data

void setup() {
      Serial.begin(9600);     // opens serial port, sets data rate to 9600 bps
}

void loop() {

      //  Data sent from the PC on Serial monitor will be echoed from the arduino and received back
      if (Serial.available() > 0) {
            // read the incoming byte:
            incomingByte = Serial.read();
            incomingByte = incomingByte - 48;
            // say what you got:
            Serial.print("I received: ");
            Serial.println(incomingByte, DEC);
      }
}
```

SoftwareSerial Library

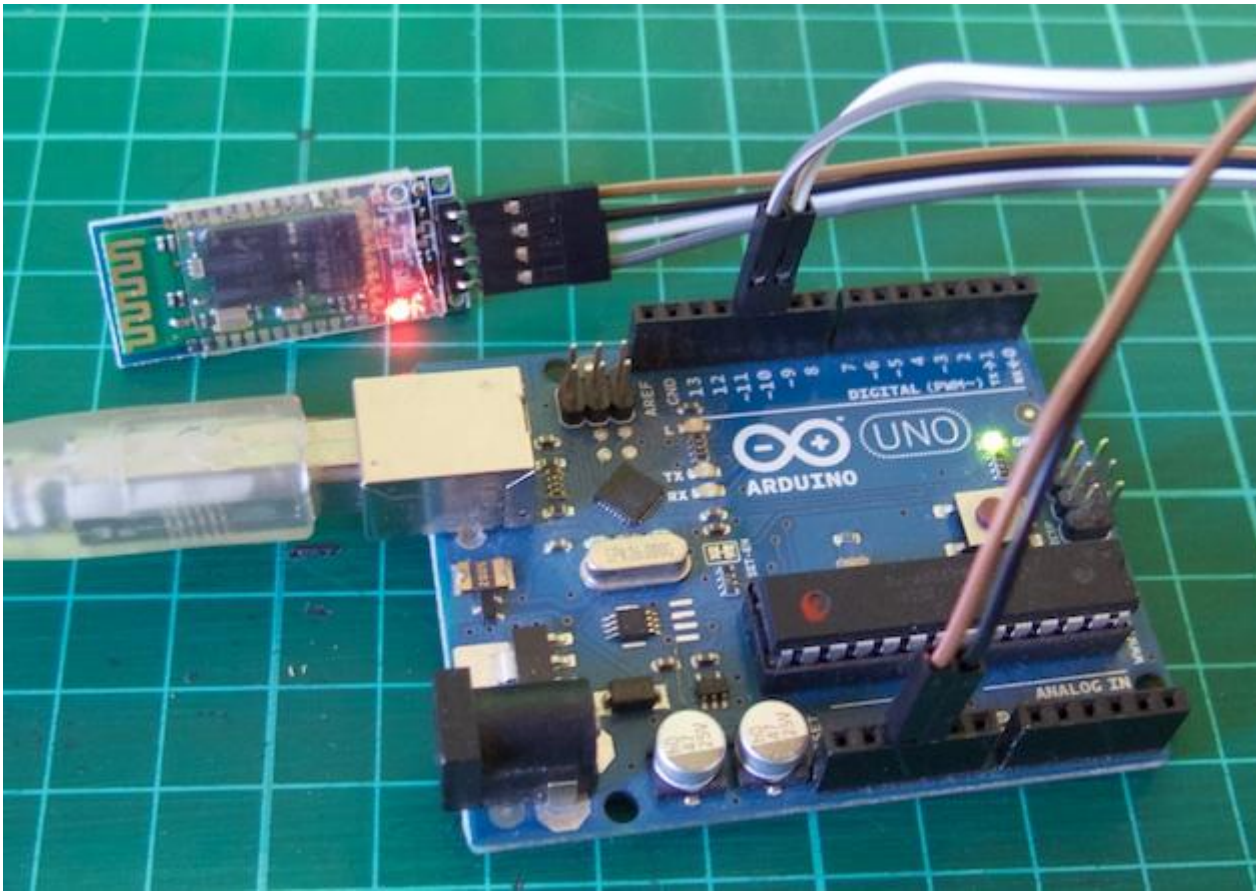
The Arduino hardware has built-in support for serial communication on pins 0 (RX) and 1(TX) (which also goes to the computer via the USB connection). The native serial support happens via a piece of hardware (built into the chip) called a UART.

The SoftwareSerial library has been developed to allow serial communication on other digital pins of the Arduino, using software to replicate the functionality (hence the name "SoftwareSerial"). It is possible to use multiple software serial ports with speeds up to 115200 bps while the native serial port on pins 0 and 1 is used for programming and serial monitor communication purposes.

The SoftwareSerial Library has been developed to allow serial communication to take place on the other digital pins of your boards, using software to replicate the functionality of the hardwired RX and TX lines. This can be extremely helpful when the need arises to communicate with two serial enabled devices, or to talk with just one device while leaving the main serial port open for debugging purpose.

Controlling Arduino using an android device via the HC06 bluetooth module and the software serial port on pins:

In the example below, digital pins 10 and 11 on your Arduino or Genuino boards are used as virtual RX and TX serial lines. The virtual RX pin is set up to listen for anything coming in on via the main serial line, and to then echo that data out the virtual TX line. Conversely, anything received on the virtual RX is sent out over the hardware TX.



Download Blueterm.apk application from Google play store on your android device and install it. You have to pair HC06 bluetooth module and your android device from the settings7bluetooth module by scanning the device and entering 12345 as the password. Once the devices are paired, you can start the BlueTerm application and connect the devices using the related menu on your android device.

Once the Bluetooth module has been connected and power applied, the LED will blink rapidly. This means that it has not been "paired" with another Bluetooth device. The LED stays on continuously when paired.

Write the following Arduino program which enables an android device to communicate with the Arduino UNO using the bluetooth device HC06 via the software serial port on pins 10 (RX) and 11 (TX). An LED connected to pin 12 can be turned ON and OFF by sending a numer 1 or 0 respectively.

```
#include <SoftwareSerial.h>
SoftwareSerial myBluetooth(10, 11); // RX, TX
int ledpin=12;
int receivedValue;
void setup() {
// put your setup code here, to run once:
Serial.begin(9600); // initialize Serial comm
Serial.println("Start");
myBluetooth.begin(9600); //initialize software serial comm
myBluetooth.println("Press 1 or 0 to Turn the LED ON or OFF");
pinMode(ledpin,OUTPUT);
}
```

```
void loop() {

if (myBluetooth.available())
{
  Serial.println("Data is received");
  myBluetooth.println("Data is received");
  receivedValue=myBluetooth.read();
  receivedValue= receivedValue-48; //convert from ASCII to decimal
if(receivedValue==1){
digitalWrite(ledpin,1);
myBluetooth.println("LED is ON");
Serial.println("LED is ON");
}
if (receivedValue== 0){
digitalWrite(ledpin,0);
myBluetooth.println("LED is OFF");
Serial.println("LED is OFF");
}
}
delay(100);// bir sonraki veriyi hazırlamak için
}
```

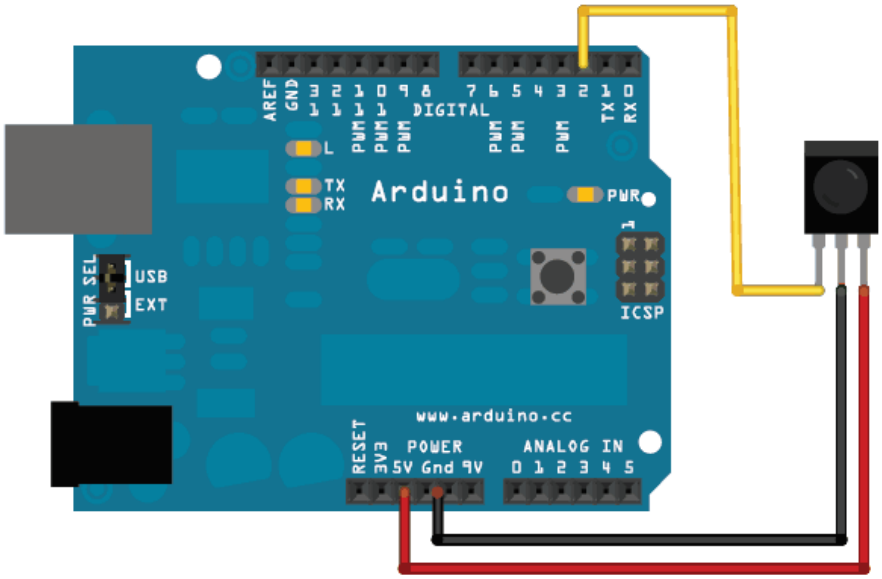
INFRARED (IR) TRANSCEİVER (REMOTE CONTROL)

Purpose: To experiment the use of an IR transceiver (remote control) module using software serial function

Procedure:

This program demonstrates receiving IR codes with IRrecv
An IR detector/demodulator must be connected to the input RECV_PIN on pin2.
Connect LED1, LED2 and LED3 on pins 8,9 and 10 respectively.
LEDS will be ON as you press the keys 1, 2 and 3.
When you press any other key, all leds will be turned OFF.

.



Write the following program in Arduino sketch and download it into Arduino UNO.

```
/*
 * This program demonstrates receiving IR codes with IRrecv
 * An IR detector/demodulator must be connected to the input RECV_PIN on pin2.
 * Connect LED1, LED2 and LED3 on pins 8,9 and 10 respectively.
 * LEDS will be ON as you press the keys 1, 2 and 3. When you press
 * any other key, all leds will be turned OFF.
 */

#include <IRremote.h>

int RECV_PIN = 2; //IR receiver module ouput is connected to pin2

IRrecv irrecv(RECV_PIN);
```

```
decode_results results;


void setup()
{
  Serial.begin(9600);
  irrecv.enableIRIn(); // Start the receiver
  pinMode(8,OUTPUT); //LED1 on pin 8
  pinMode(9, OUTPUT); //LED2 on pin 9
  pinMode(10, OUTPUT); //LED3 on pin 10
}

void loop() {
  if (irrecv.decode(&results)) {
  switch (results.value) {
  case 0xFF30CF:  // If 1 is pressed do the following
    digitalWrite(8,1); // Turn ON LED1
    digitalWrite(9,0); // Turn OFF LED2
    digitalWrite(10,0); // Turn ON LED3
    Serial.println("LED1 is ON");
    break;
   case 0xFF18E7:  // If 1 is pressed do the following
    digitalWrite(8,0); // Turn ON LED1
    digitalWrite(9,1); // Turn OFF LED2
    digitalWrite(10,0); // Turn ON LED3
    Serial.println("LED2 is ON");
    break;
     case 0xFF7A85:  // If 1 is pressed do the following
    digitalWrite(8,0); // Turn ON LED1
    digitalWrite(9,0); // Turn OFF LED2
    digitalWrite(10,1); // Turn ON LED3
    Serial.println("LED3 is ON");
    break;
  default: //If any other pin is pressed
    digitalWrite(8,0); // Turn ON LED1
    digitalWrite(9,0); // Turn OFF LED2
    digitalWrite(10,0); // Turn ON LED3
    Serial.println("ALL LEDS ARE OFF");
  break;
}

    Serial.println(results.value, HEX);
    irrecv.resume(); // Receive the next value
   }
    delay(500);
}
```

millis() : TIMER0 application

Description

Returns the number of milliseconds since the Arduino board began running the current program. This number will overflow (go back to zero), after approximately 50 days. You can use this function in your timing applications.

Returns

Number of milliseconds since the program started (*unsigned long*)

Note:

Please note that the return value for millis() is an unsigned long, logic errors may occur if a programmer tries to do arithmetic with smaller data types such as int's. Even signed long may encounter errors as its maximum value is half that of its unsigned counterpart.

Example

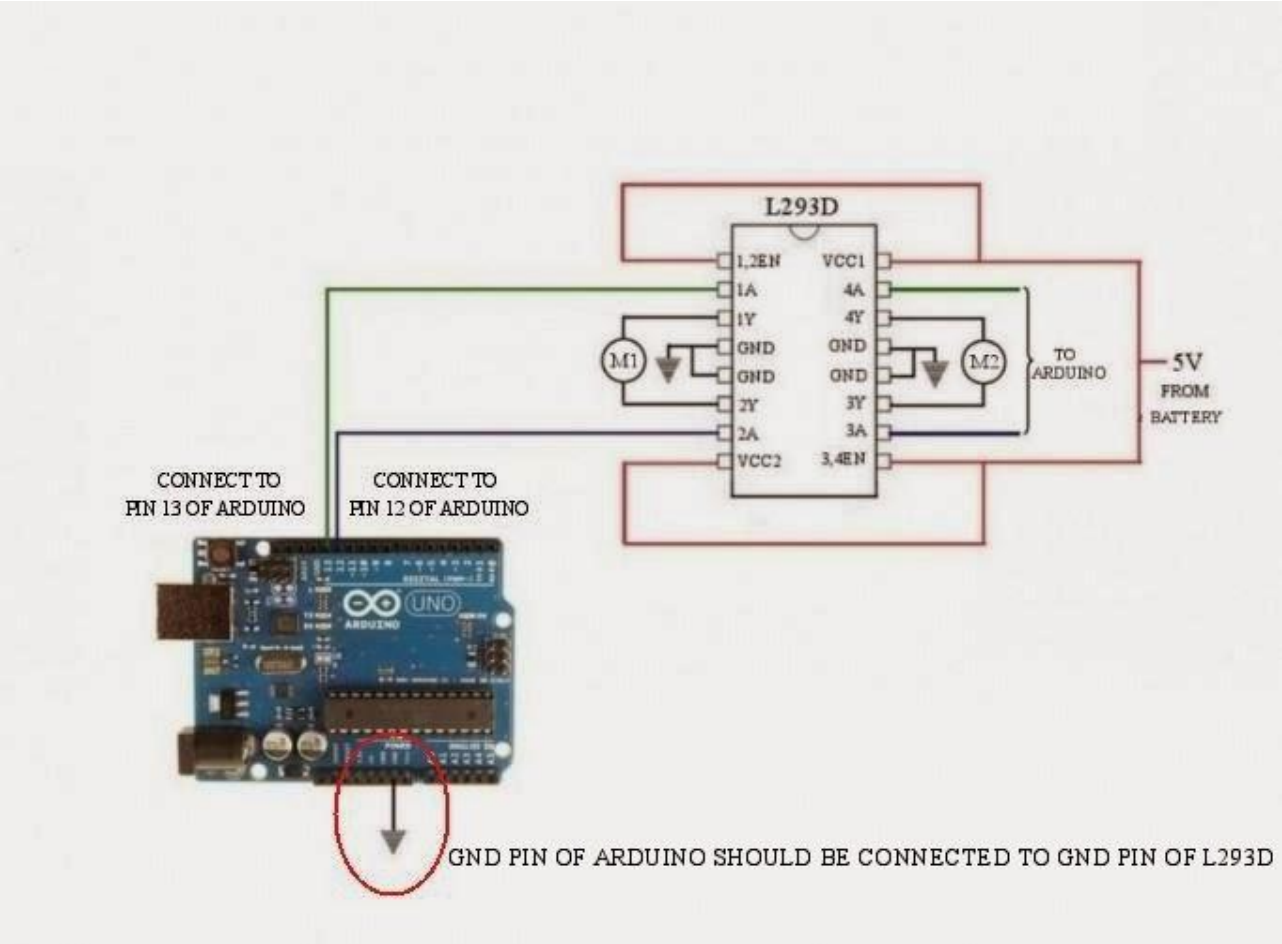
```
unsigned long time;

void setup(){
  Serial.begin(9600);
}
void loop(){
  Serial.print("Time: ");
  time = millis();
  //prints time since program started
  Serial.println(time);
  // wait a second so as not to send massive amounts of data
  delay(1000);
}
```

BUTTON & INTERRUPT

```
/* This program demonstrates attaching an interrupt to arduino UNO.
 *  A button is connected to pin 2 (interrupt input 0) and an LED
 *  is connected to pin 6. Each time the button is pressed, an
 *  interrupt is generated due to the state change on pin2 and the
 *  program jumps to the interrupt routine called "reverse" to reverse
 *  state of the LED on pin 6.

*/
int state = 0; // initialize the variable state as 0
void setup() {
   pinMode(6, OUTPUT);
  attachInterrupt(0, reverse, CHANGE); //Interrupt will occur when the state of pin 2 changes
  digitalWrite(6,state); // turn the LED on 6 OFF

}

void loop() {
 //wait until a button on pin 2 is pressed
}
void reverse()  // interrupt routine to be executed when a change occurs on pin 2
{
   state = ~state;
   digitalWrite(6,state); // turn the LED ON or OFF according to the new state

}
```

Control DC Motor Direction Using L293D Motor Driver and Arduino

L293D is a popular motor driver. It is special by its high current and voltage rating. Each L293D can be used to control two motors. Each side controls each motor. 1A and 2A is the control pins for motor M1. Similarly 3A and 4A is the control pins for motor M2. Here, the circuit is demonstrated to control M1 only. Control pins are normally connected to Microcontrollers. I used Arduino to give control signals. Circuit Diagram is as shown in figure given below.



Direction of rotation as per the control signals is represented in the following table.

1A (PIN 2 OF L293D)	2A (PIN 7 OF L293D)	M1
1	1	-
1	0	CLOCKWISE
0	1	ANTICLOCKWISE
0	0	-

```
/*

 * DC motor speed and direction control using L293D HBridge IC. Pin 8 and Pin 9
 * are used as the control pins. When pin 8 is made zero and a PWM is applied to
 * pin 9, motor rotates in clockwise direction and the speed is controlled by
 * changing the duty ratio which is defined by the potentiometer output.
 * Direction can be reversed by changing the polarity of the voltage by changing the
 * pin numbers.
 */
void setup()
{
  pinMode(8, OUTPUT); //8th pin is used as the zero reference terminal
  pinMode(9, OUTPUT);   // PWM voltage is produced on the 9th pin to control the speed.
  Serial.begin(9600);
 }
void loop()
{
 digitalWrite(8,HIGH); // Pin 8 is set to zero
  int analogDeger = analogRead(A0);   // Potentiometer value is read to determine the duty ratio
 int Duty = analogDeger/4;  // Raw pot value 0 - 1023 is converted to the duty values 0 - 255
 analogWrite(9, Duty);  // PWM is generated on pin 9 according to the pot value
 Serial.println(Duty);
 delay(100) ; //wait for 100 ms
}
```

Exercise 3 – Basic I/O: Light Level Indicator with Light Dependent Resistor

A. Objectives

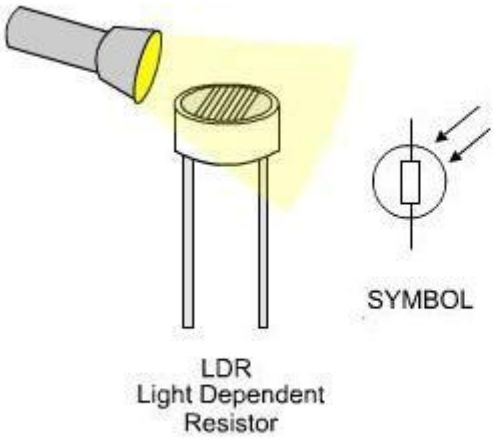
This exercise aims for the student to apply the basic I/O methods learned from the previous experiments with the ADC. As such, the exercise will also serve as an introductory to Analog-to-Digital Conversion of the Arduino Uno. Here, a Light level indicator will be made with an LDR. By the end of this exercise, the student should have a good grasp of the I/O features of the board and understand the ADC function of the Analog Input pins.

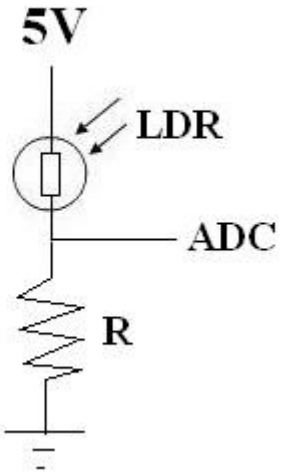
B. Questions

- How does the LDR Work? Does the resistance increase as it receives more light?
- What is the necessary function to read analog inputs?
- How does the ADC of the analog input pins process the voltages it receives?
- What estimated range values is needed for the program to determine the light level?

C. Materials Needed

- 1 x Arduino Uno board and USB cable
- 4 x 5mm/3mm LEDs
- 4 x 470Ω resistors and 1 x 2.2kΩ (= R) resistor
- 1 x breadboard
- 1 x Light Dependent Resistor (LDR)





D. Introduction to Arduino Uno's Analog-to-Digital Converter

The Arduino Uno board has exactly 6 pins that can read analog values. With the analog pins, you process values with more significant digits that are needed in applications.

E. Instructions to Programming the LED lamp using an LDR as a switch

1. Open Arduino, and create first the part that would read the analog input coming from one of the analog input pins.
2. Now, construct a conditional statement for the ranges wherein the number of LEDs turned ON correspond to how less the light received is. This means 4 LEDs light up in a dark room while none in a well lit room. *Note to study the behavior of an LDR and determine the relationship of the resistance and the light it receives*. *Expect some calibrating to be done.* Also, remember that the LDR is connected to the builtin 5V source. The less the LDR's resistance the higher the voltage the analog pins receive.
3. Once you are done writing the code, save your file and compile it.

F. Test your program

1. Construct the circuit using the Arduino Uno, breadboard, LED, resistors, and LDR.
2. Connect the board to the PC using the USB cable. Upload the sketch saved earlier and check if the program works. The 4 LEDs should light up in a dark room while none in a well lit room. *If it did not work, some calibration to the ranges specified might be needed.*

G. What to expect

You have just written a sketch and programmed the Arduino Uno board to control an LED light level indicator based on the amount of light the sensor (LDR) receives. After covering the LDR with an object or your hand, you should see the 4 LEDs turn ON and upon uncovering, the number of lit LED's decrease.

H. Summary

In this exercise, you have learned how to use the Arduino Board and IDE to control outputs based on results from a sensor. You have also learned how the ADC feature of the analog input pins works by controlling the number of LED's turned ON based on the amount of light the LDR receives.

ARDUINO BASED INSTRUMENTATION KIT
EDS01 USER MANUAL

Arduino based Instrumentation Kit is composed of an Arduino Uno card, sensors, displays, buttons, optical isolators, motor control circuits and a relay to teach:

1. Basic programming techniques of microcontrollers,
2. Connections and the configuration of sensors to receive information from the physical world,
3. How to communicate and control the devices with an android device or with a laptop via the wireles modules,
4. How to monitor the variables involved in the experiment,
5. How to control the speed and position of DC, step and servo motors
6. How to control the power of external devices such as heaters which are supplied with 220 V line voltage using a contactor,
7. How to us optical isolators to isolate the micrcontroller output terminals from the devices connected to the 220 V line voltage.

Grounds of all devices are connected to a common point, hence no ground connection is required during the experimental set up.

5V Vcc supply of the sensors and motors are connected by Dip Switches on SWDIP5. The data connections of the sensors and displays can be made by using the single female to female jumpers or alternatively by using the dip switches as shown below.