

Abstract

In common every person able to measure the speed of a vehicle using a speedometer. But it is simply a question mark when we are trying to find out the speed of an action which ends up within micro second. For this scenario we introduced an instrument named Speede which can measure the speed for our intended actions.

Introduction

This paper is dealing with **SPEEDE** an instrument that can measure the speed of any object that passes through the instrument (provided it crosses the two lasers).

Tools Required

- Arduino Board
- Wooden board

Software Required

- Arduino 1.0.6

Components Required

- Laser - 2
- LDR (Light Detecting Resistor) - 2
- Resistors ($1k\Omega$) - 2
- Liquid Crystal Display (JHD162A)
- Printed Circuit Board (PCB) – 2
- Potentiometer ($1k\Omega$)

Arduino Board

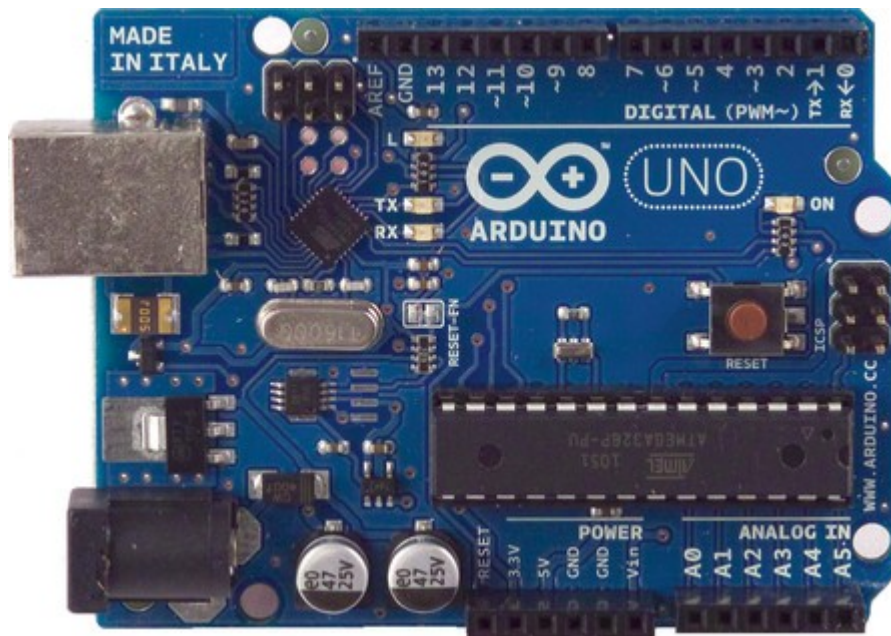
The Arduino Uno is a microcontroller board based on the ATmega328. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.

Summary:

Microcontroller	ATmega328
Operating Voltage	5V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V

The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega16U2 (Atmega8U2 up to version R2) programmed as a USB-to-serial converter.

ARDUINO BOARD



Microcontroller

Arduino is a microcontroller on a circuit board which makes it easy to receive inputs and drive outputs. A microcontroller is a integrated computer on a chip.

Inputs: Some exaples of inputs would be a temperature sensor, a motion sensor, a distance sensor, a switch and so fort

In this project we make use of 10 pins of Arduino as described follows,

Arduino pin	To PCB
GND	Ground
+5V	VCC
1 (Digital)	DB 1
2(Digital)	DB2
3 (Digital)	DB3
4 (Digital)	DB4
5(Digital)	Enable
6 (Digital)	Register Select
12 (Digital)	LDR stop
13 (Digital)	LDR start

Outputs: Some examples of outputs would be a light, a screen, a motor and so forth.

TL;DR: Arduino is a small computer that you can program to read and control electrical components connected to it.

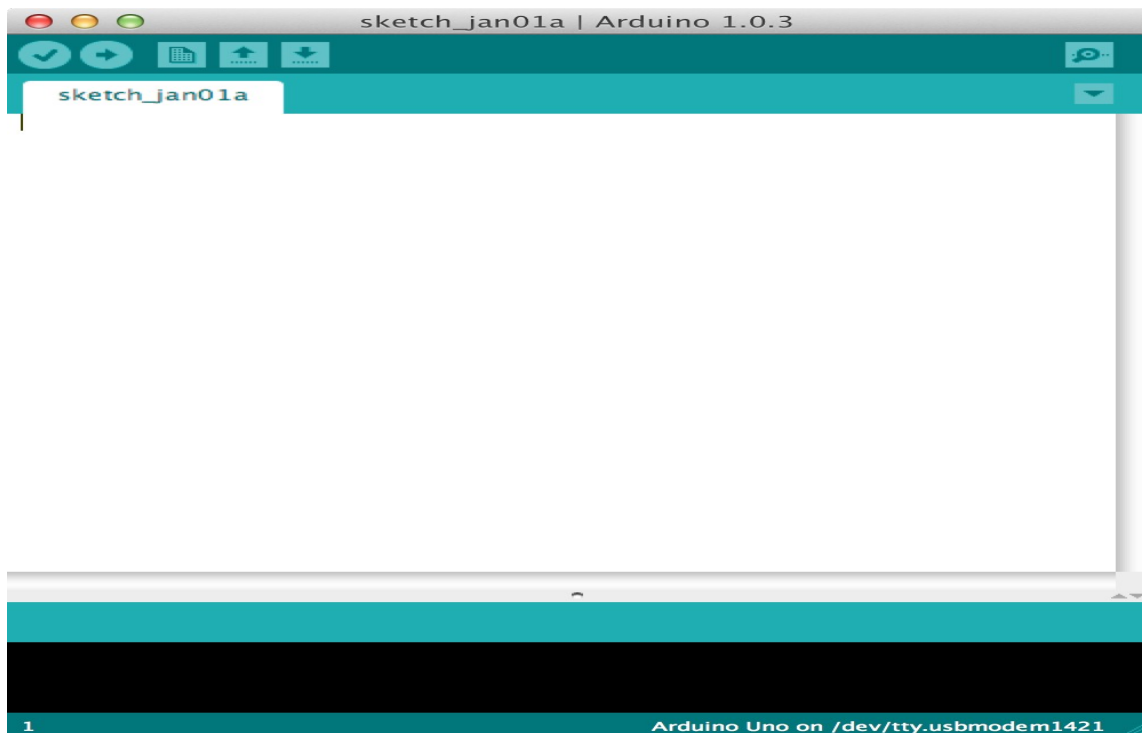
Arduino do not Read the resistance value. So here we used voltage divider as the part of the connection.

Wooden Board

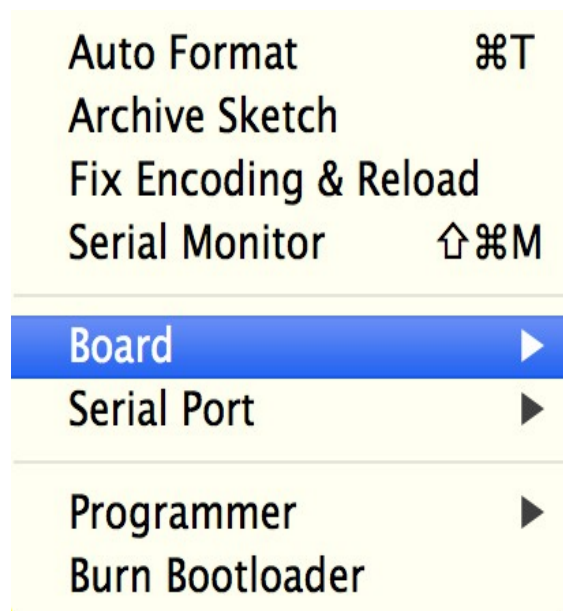
he purpose of the wooden board is to hold the circuits and laser pointers.

Arduino 1.0.6

You'll need to download the Arduino Software package for your operating system. When you've downloaded and opened the application you should see something like this:



This is where you type the code you want to compile and send to the Arduino board.



The Initial Setup:

We need to setup the environment to **Tools** menu and select **Board**.

Then select the type of Arduino you want to program, in our case it's the **Arduino Uno**.

✓ Arduino Uno

- Arduino Duemilanove w/ ATmega328
- Arduino Diecimila or Duemilanove w/ ATmega168
- Arduino Nano w/ ATmega328
- Arduino Nano w/ ATmega168
- Arduino Mega 2560 or Mega ADK
- Arduino Mega (ATmega1280)
- Arduino Leonardo
- Arduino Esplora
- Arduino Micro
- Arduino Mini w/ ATmega328
- Arduino Mini w/ ATmega168
- Arduino Ethernet
- Arduino Fio
- Arduino BT w/ ATmega328
- Arduino BT w/ ATmega168
- LilyPad Arduino USB
- LilyPad Arduino w/ ATmega328
- LilyPad Arduino w/ ATmega168
- Arduino Pro or Pro Mini (5V, 16 MHz) w/ ATmega328
- Arduino Pro or Pro Mini (5V, 16 MHz) w/ ATmega168
- Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega328
- Arduino Pro or Pro Mini (3.3V, 8 MHz) w/ ATmega168
- Arduino NG or older w/ ATmega168
- Arduino NG or older w/ ATmega8

The Code:

The code you write for your Arduino are known as **sketches**. They are written in **C++**.

Every sketch needs two *void type functions*, `setup()` and `loop()`. A void type function doesn't return any value.

The `setup()` method is ran once at the just after the Arduino is powered up and the `loop()` method is ran continuously afterwards. The `setup()` is where you want to do any initialisation steps, and in `loop()` you want to run the code you want to run over and over again.

So, your basic sketch or program should look like this:

```

void setup()
{

}

void loop()
{

}

```

In our code above the setup() method let's create a variable called const int LDR_start & const int LDR_stop. In C++ we need to state why type our variable is before hand, in this case it's an integer, so it's of type int.

Here is the program which dumped in the Arduino board.

```

/*
Timing Measurement
The circuit:
* LDR_start attached from pin 13 to ground
* LDR_stop attached from pin 12 to ground
* The circuit:
* LCD RS pin to digital pin 1
* LCD Enable pin to digital pin 2
* LCD D4 pin to digital pin 3
* LCD D5 pin to digital pin 4
* LCD D6 pin to digital pin 5
* LCD D7 pin to digital pin 6
* LCD R/W pin to ground
  • LCD 1st & 5th to VCC (+5V) and 3rd to ground

*/
// include the library code:
#include <LiquidCrystal.h>

// constants won't change. They're used here to
// set pin numbers:
const int LDR_start = 13; // the number of the pushbutton pin
const int LDR_stop = 12; // the number of the LED pin

// Some variables to read the input digital information
int start_var = 0;
int stop_var = 0;

// variables will change:
int State = 0; // variable for reading the pushbutton status
// 0 is waiting/armed
// 1 is time started
// 2 is time stop

```

```

// Start and stop times
unsigned long start_time_us;
unsigned long stop_time_us;
float delta_time;
float speed_var;

// initialize the library with the numbers of the interface pins
LiquidCrystal lcd(1, 2, 3, 4, 5, 6);

void setup() {
  // initialize the LDR_start:
  pinMode(LDR_start, INPUT);
  // initialize the LDR_stop:
  pinMode(LDR_stop, INPUT);
  // set up the LCD's number of columns and rows:
  lcd.begin(16, 2);
  lcd.clear();
  lcd.setCursor(0, 0);

  lcd.print("Speede is ready");// Print a message to the LCD.
}

void loop(){
  // Serial.print("State = ");
  // Serial.println(State);
  switch(State){
    case 0:
      // 0 is waiting/armed
      start_var = digitalRead(LDR_start);
      if(start_var == 1){
        start_time_us = micros();
        State = 1;
      }
      break;
    case 1:
      stop_var = digitalRead(LDR_stop);
      if(stop_var == 1){
        stop_time_us = micros();
        State = 2;
      }
      break;
    default:
      lcd.clear();
      lcd.setCursor(0, 0);
      delta_time = (stop_time_us - start_time_us)/1e6;
      lcd.print("Time Delta in s");
      lcd.setCursor(0,1);
      lcd.print(delta_time);
      delay(3000);

      speed_var = 5.5e-2/delta_time;
      lcd.clear();

```

```

    lcd.setCursor(0, 0);
    lcd.print("Speed in m/s");
    lcd.setCursor(0,1);
    lcd.print(speed_var);
    delay(3000);
    lcd.clear();

    speed_var = speed_var*3.6;
    lcd.clear();
    lcd.setCursor(0, 0);
    lcd.print("Speed in km/hr");
    lcd.setCursor(0,1);
    lcd.print(speed_var);
    delay(3000);

}
}

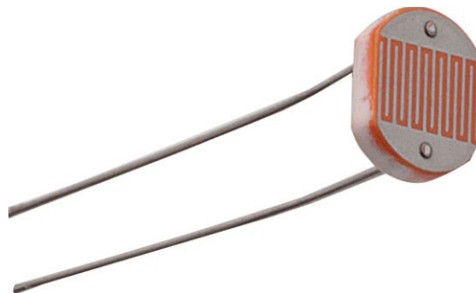
```

If encounter with the error “ serial port '/dev/ttyACM0' not found” just execute “sudo chmod a+rw /dev/ttyACM0” command in the terminal.

Each line is ended with a semicolon (;).

Light Dependent Resisiter

A photoresistor or light-dependent resistor (LDR) or photocell is a light-controlled variable resistor. The resistance of a photoresistor decreases with increasing incident light intensity; in other words, it exhibits photoconductivity.



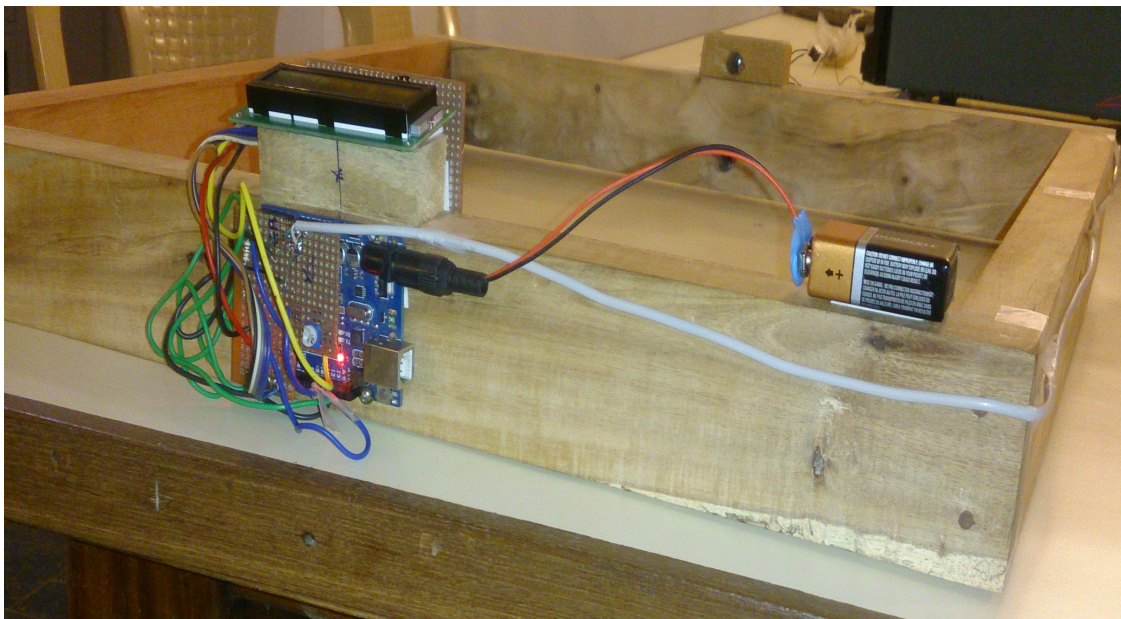
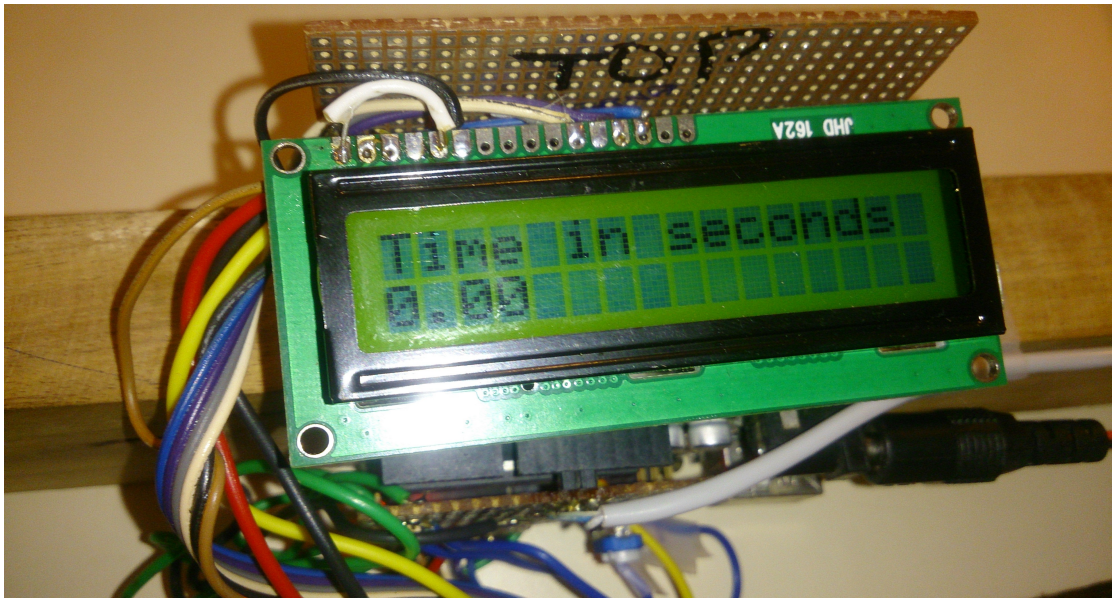
Set up & Working

Two Light Dependent Resistors (LDR) were positioned just opposite to two Laser pointers in a printed circuit board in order to vary the resistivity of the LDRs accordance with the lasers get ON and OFF.

Here the purpose of putting two photo resistors is to detect the time interval of the obstacle which prevent the laser to hit the LDR. As mentioned above when the laser hit the resister its resistivity goes down upto some point & returns back to the original value when the light source gets switched off. By measuring the time interval between the 1st LDR that changed its resistivity and the 2nd LDR that returned back to its original resistivity, one can determine the time interval taken by the obstacle at that instant.

Liquid Crystal Display

Liquid Crystal Display JHD162A used to display the output of SPEEDE. That is, this set up helps to show the time period of the action and speed in m/s & km/h.



Pin Diagram of LCD

Pin	Symbol	I/O	Description
1	VSS	-	Ground
2	VCC	-	+5v power supply
3	VEE	-	Control contrast
4	RS	I	RS=0 to select command Register, RS=1 to select data register
5	R/W	I	R/W=0 for write, R/W=1 for read
6	E	I/O	Enable
11	DB1	I/O	Data bus
12	DB2	I/O	Data bus
13	DB3	I/O	Data bus
14	DB4	I/O	Data bus

LCD Set up:

The first thing is to do is what figure out the LCD pins to be connected with the Arduino board to write the output in the LCD as shown in figure below.

- VSS & VCC are connected to arduino board's ground & 5v supply pins.
- RS, E, DB1, DB2, DB3, DB4 are connected to 6,5,4,3,2,1 (digital pins of arduino).
- VEE connected to the potentiometer.

Speede:

After integrating all the components in their position as per the design, Speede is ready. This gives the time period of an action, speed in m/s & km/h.

REFERENCE

<https://www.google.co.in>

<http://www.circuitstoday.com/interfacing-lcd-to-arduino>

<http://www.technologystudent.com/elec1/ldr1.htm>