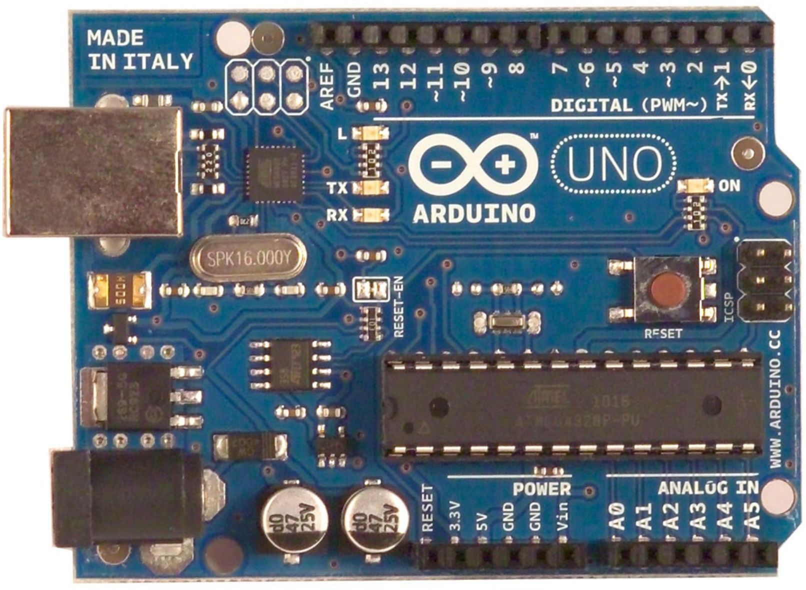
LAMPIRAN

Lampiran Listing Program Arduino

```
#include <Wire.h> // I2C library, required for MLX90614
#include <LiquidCrystal_I2C.h>
#define I2C_ADDR 0x27 //I2C adress, you should use the code to scan the adress first (0x27) here
#define BACKLIGHT_PIN 3 // Declaring LCD Pins
#define En_pin 2
#define Rw_pin 1
#define Rs_pin 0
#define D4_pin 4
#define D5_pin 5
#define D6_pin 6
#define D7_pin 7
LiquidCrystal_I2C lcd(I2C_ADDR,En_pin,Rw_pin,Rs_pin,D4_pin,D5_pin,D6_pin,D7_pin);

int relayfan = 12;
int relaysedot = 11;
int relayuap = 10;
int smokeA1 = A1;
int smokeA2 = A2;

void setup() {
  Serial.begin(9600);
  lcd.begin(16,2);
    lcd.setBacklightPin(BACKLIGHT_PIN,POSITIVE);
  lcd.setBacklight(HIGH); //Lighting backlight

  pinMode(relayfan, OUTPUT);
  pinMode(relaysedot, OUTPUT);
  pinMode(relayuap , OUTPUT);
  pinMode(smokeA1, INPUT);
  pinMode(smokeA2, INPUT);

digitalWrite(relayuap , HIGH);
digitalWrite(relaysedot, HIGH);
digitalWrite(relayfan, HIGH);

}

void loop() {
  int analogSensor = analogRead(smokeA1);
  int analogSensor1 = analogRead(smokeA2);
  Serial.print("Pin A1: ");
  Serial.println(analogSensor);
  Serial.print("Pin A2: ");
  Serial.println(analogSensor1);

 lcd.setCursor(0,0);
 lcd.print("Sensor 1: ");
 lcd.print(analogSensor);
 lcd.print("PPM");

 lcd.setCursor(0,1);
 lcd.print("sensor 2: ");
 lcd.print(analogSensor1);
 lcd.print("PPM");

  // Checks if it has reached the threshold value
  if ((analogSensor1 >= 160)||(analogSensor >= 160))
  {Serial.print("fan hidup: ");
    digitalWrite(relaysedot, LOW);
    digitalWrite(relayfan, LOW);
     digitalWrite(relayuap , LOW);

  }
  else if ((analogSensor1 >= 120 && analogSensor1 <= 160 )||(analogSensor >= 120 && analogSensor
<= 160 ))
  {Serial.print("uap hidup: ");
    digitalWrite(relayuap , LOW);
    digitalWrite(relaysedot, LOW);
    digitalWrite(relayfan, HIGH);

  }
else if ((analogSensor1 <= 120)||(analogSensor <= 120))
  {
    digitalWrite(relayuap , HIGH);
    digitalWrite(relaysedot, HIGH);
    digitalWrite(relayfan, HIGH);

  }
  delay(500);
}
```

Arduino UNO



Product Overview

The Arduino Uno is a microcontroller board based on the ATmega328 (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz crystal oscillator, 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. The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega8U2 programmed as a USB-to-serial converter.

"Uno" means one in Italian and is named to mark the upcoming release of Arduino 1.0. The Uno and version 1.0 will be the reference versions of Arduno, moving forward. The Uno is the latest in a series of USB Arduino boards, and the reference model for the Arduino platform; for a comparison with previous versions, see the index of Arduino boards.

Index

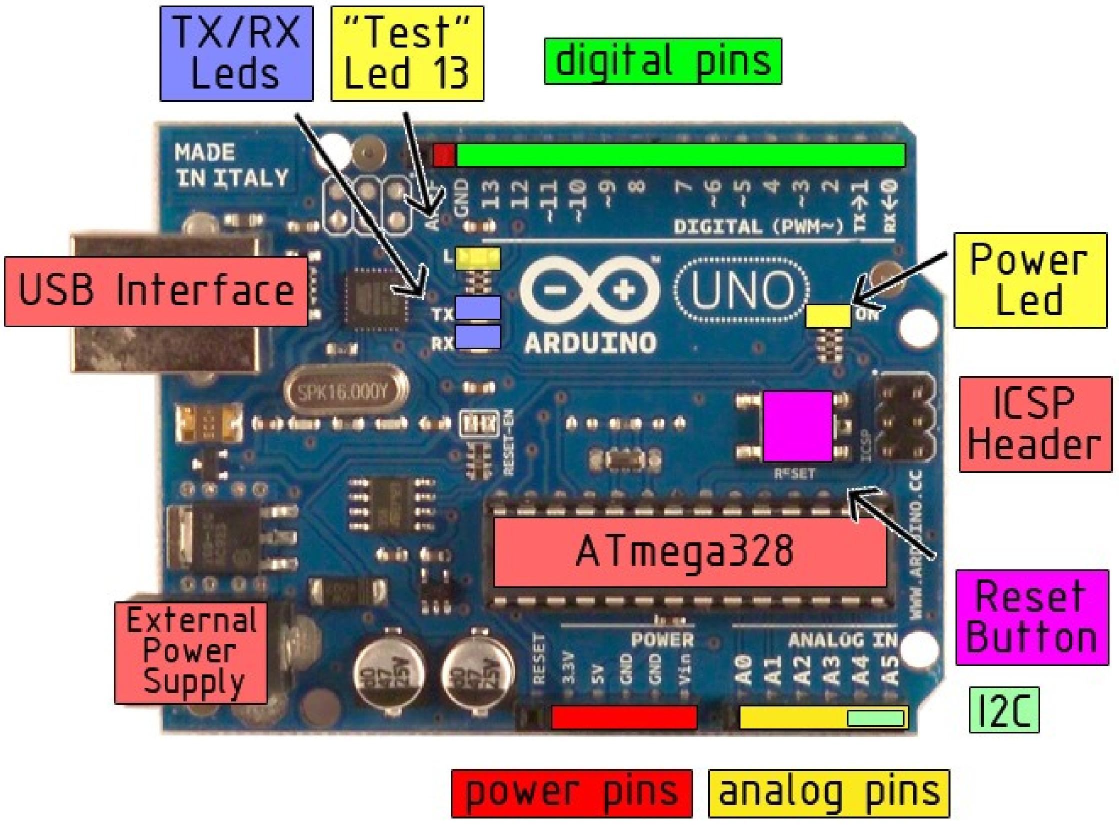
Technical Specification

EAGLE files: arduino-duemilanove-uno-design.zip Schematic: arduino-uno-schematic.pdf

Summary

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
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
Flash Memory	32 KB of which 0.5 KB used by bootloader
SRAM	2 KB
EEPROM	1 KB
Clock Speed	16 MHz

the board



Power

The Arduino Uno can be powered via the USB connection or with an external power supply. The power source is selected automatically.

External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.

The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The power pins are as follows:

- **VIN.** The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- **5V.** The regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5V supply.
- **3V3.** A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- **GND.** Ground pins.

Memory

The Atmega328 has 32 KB of flash memory for storing code (of which 0,5 KB is used for the bootloader); It has also 2 KB of SRAM and 1 KB of EEPROM (which can be read and written with the EEPROM library).

Input and Output

Each of the 14 digital pins on the Uno can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

- **Serial: 0 (RX) and 1 (TX).** Used to receive (RX) and transmit (TX) TTL serial data. TThese pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip .
- **External Interrupts: 2 and 3.** These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.
- **PWM: 3, 5, 6, 9, 10, and 11.** Provide 8-bit PWM output with the analogWrite() function.
- **SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK).** These pins support SPI communication, which, although provided by the underlying hardware, is not currently included in the Arduino language.

- **LED: 13.** There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

The Uno has 6 analog inputs, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and the analogReference() function. Additionally, some pins have specialized functionality:

- **I^{2}C: 4 (SDA) and 5 (SCL).** Support I^{2}C (TWI) communication using the Wire library.

There are a couple of other pins on the board:

- **AREF.** Reference voltage for the analog inputs. Used with analogReference().
- **Reset.** Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

See also the mapping between Arduino pins and Atmega328 ports.

Communication

The Arduino Uno has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega8U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The '8U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, on Windows, an *.inf file is required..

The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

A SoftwareSerial library allows for serial communication on any of the Uno's digital pins.

The ATmega328 also support I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the documentation for details. To use the SPI communication, please see the ATmega328 datasheet.

Programming

The Arduino Uno can be programmed with the Arduino software (download). Select "Arduino Uno w/ ATmega328" from the **Tools > Board** menu (according to the microcontroller on your board). For details, see the reference and tutorials.

The ATmega328 on the Arduino Uno comes preburned with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol (reference, C header files).

You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header; see these instructions for details.

The ATmega8U2 firmware source code is available . The ATmega8U2 is loaded with a DFU bootloader, which can be activated by connecting the solder jumper on the back of the board (near the map of Italy) and then resetting the 8U2. You can then use Atmel's FLIP software (Windows) or the DFU programmer (Mac OS X and Linux) to load a new firmware. Or you can use the ISP header with an external programmer (overwriting the DFU bootloader).

Automatic (Software) Reset

Rather than requiring a physical press of the reset button before an upload, the Arduino Uno is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the ATmega8U2 is connected to the reset line of the ATmega328 via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino software uses this capability to allow you to upload code by simply pressing the upload button in the Arduino environment. This means that the bootloader can have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload.

This setup has other implications. When the Uno is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following half-second or so, the bootloader is running on the Uno. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened. If a sketch running on the board receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data.

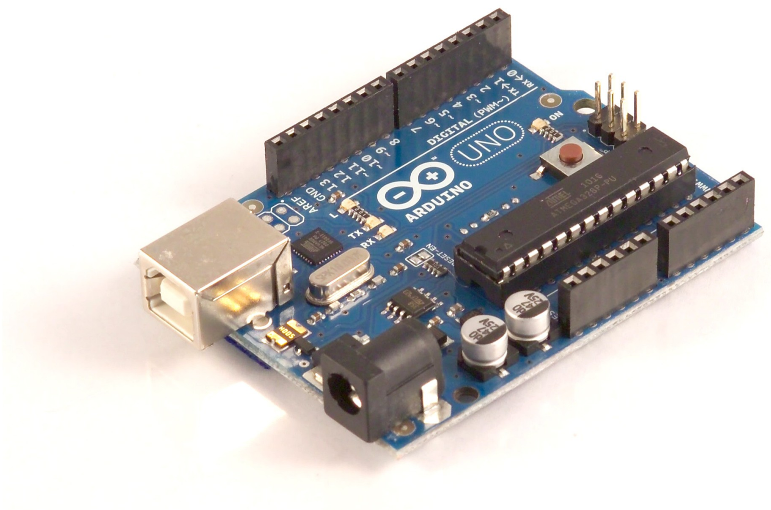
The Uno contains a trace that can be cut to disable the auto-reset. The pads on either side of the trace can be soldered together to re-enable it. It's labeled "RESET-EN". You may also be able to disable the auto-reset by connecting a 110 ohm resistor from 5V to the reset line; see this forum thread for details.

USB Overcurrent Protection

The Arduino Uno has a resettable polyfuse that protects your computer's USB ports from shorts and overcurrent. Although most computers provide their own internal protection, the fuse provides an extra layer of protection. If more than 500 mA is applied to the USB port, the fuse will automatically break the connection until the short or overload is removed.

Physical Characteristics

The maximum length and width of the Uno PCB are 2.7 and 2.1 inches respectively, with the USB connector and power jack extending beyond the former dimension. Three screw holes allow the board to be attached to a surface or case. Note that the distance between digital pins 7 and 8 is 160 mil (0.16"), not an even multiple of the 100 mil spacing of the other pins.



How to use Arduino

Arduino can sense the environment by receiving input from a variety of sensors and can affect its surroundings by controlling lights, motors, and other actuators. The microcontroller on the board is programmed using the Arduino programming language (based on Wiring) and the Arduino development environment (based on Processing). Arduino projects can be stand-alone or they can communicate with software on running on a computer (e.g. Flash, Processing, MaxMSP).

Arduino is a cross-platoform program. You'll have to follow different instructions for your personal OS. Check on the Arduino site for the latest instructions. *http://arduino.cc/en/Guide/HomePage*

Linux Install

Windows Install

Mac Install

Once you have downloaded/unzipped the arduino IDE, you can Plug the Arduino to your PC via USB cable.

Blink led

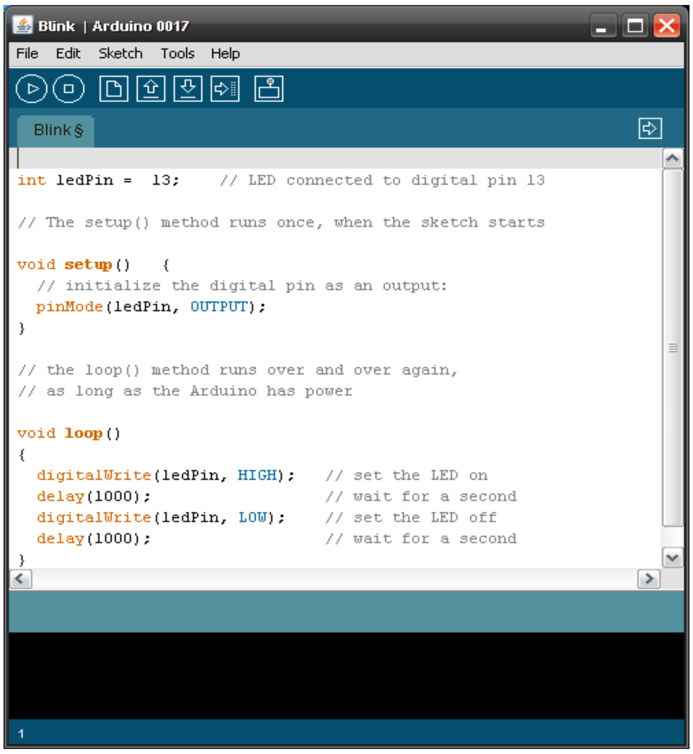
Now you're actually ready to "burn" your first program on the arduino board. To select "blink led", the physical translation of the well known programming "hello world", select

File>Sketchbook>
Arduino-0017>Examples>
Digital>Blink

Once you have your skecth you'll see something very close to the screenshot on the right.

In **Tools>Board** select

Now you have to go to
Tools>SerialPort
and select the right serial port, the one arduino is attached to.



```
int ledPin =  13;    // LED connected to digital pin 13

// The setup() method runs once, when the sketch starts

void setup()   {
  // initialize the digital pin as an output:
  pinMode(ledPin, OUTPUT);
}

// the loop() method runs over and over again,
// as long as the Arduino has power

void loop()
{
  digitalWrite(ledPin, HIGH);   // set the LED on
  delay(1000);                  // wait for a second
  digitalWrite(ledPin, LOW);    // set the LED off
  delay(1000);                  // wait for a second
}
```

Done compiling.

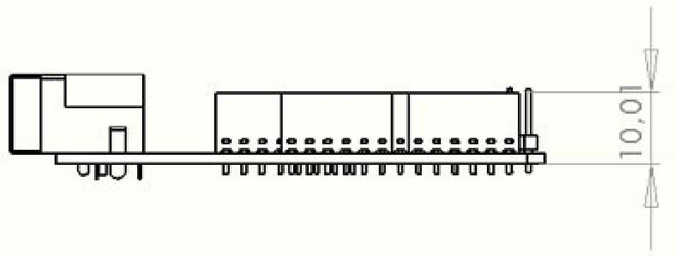
Press Compile button
(to check for errors)

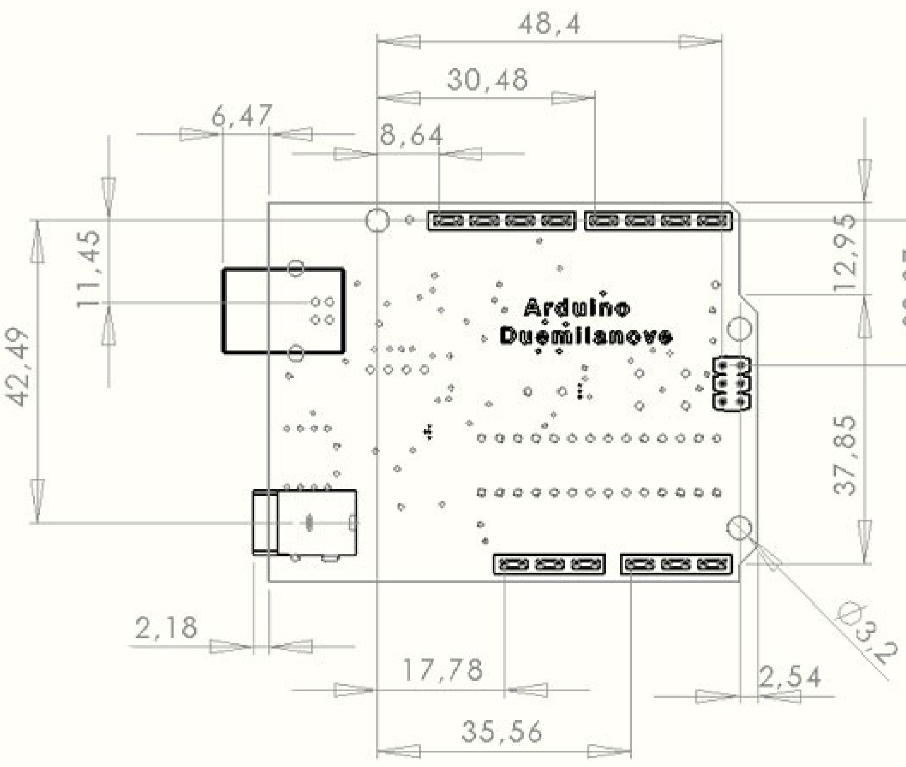
Upload

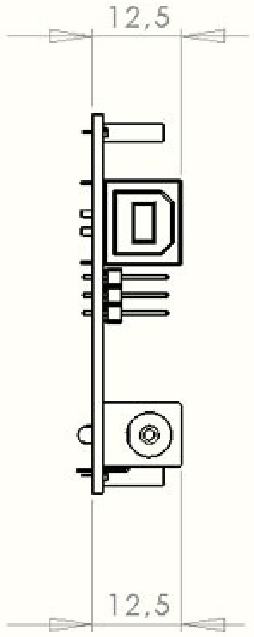
TX RX Flashing

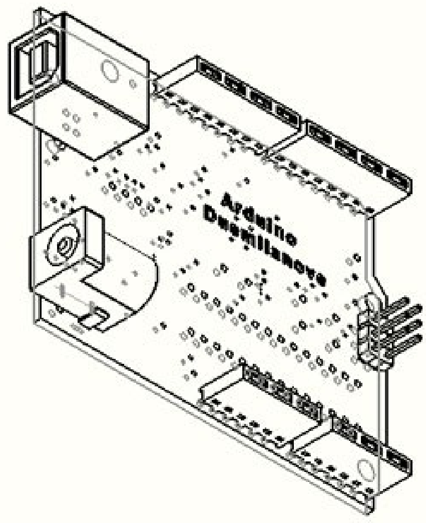
Blinking Led!

Dimensioned Drawing









Terms & Conditions

1. Warranties

1.1 The producer warrants that its products will conform to the Specifications. This warranty lasts for one (1) years from the date of the sale. The producer shall not be liable for any defects that are caused by neglect, misuse or mistreatment by the Customer, including improper installation or testing, or for any products that have been altered or modified in any way by a Customer. Moreover, The producer shall not be liable for any defects that result from Customer's design, specifications or instructions for such products. Testing and other quality control techniques are used to the extent the producer deems necessary.

1.2 If any products fail to conform to the warranty set forth above, the producer's sole liability shall be to replace such products. The producer's liability shall be limited to products that are determined by the producer not to conform to such warranty. If the producer elects to replace such products, the producer shall have a reasonable time to replacements. Replaced products shall be warranted for a new full warranty period.

1.3 EXCEPT AS SET FORTH ABOVE, PRODUCTS ARE PROVIDED "AS IS" AND "WITH ALL FAULTS." THE PRODUCER DISCLAIMS ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, REGARDING PRODUCTS, INCLUDING BUT NOT LIMITED TO, ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE

1.4 Customer agrees that prior to using any systems that include the producer products, Customer will test such systems and the functionality of the products as used in such systems. The producer may provide technical, applications or design advice, quality characterization, reliability data or other services. Customer acknowledges and agrees that providing these services shall not expand or otherwise alter the producer's warranties, as set forth above, and no additional obligations or liabilities shall arise from the producer providing such services.

1.5 The Arduino™ products are not authorized for use in safety-critical applications where a failure of the product would reasonably be expected to cause severe personal injury or death. Safety-Critical Applications include, without limitation, life support devices and systems, equipment or systems for the operation of nuclear facilities and weapons systems. Arduino™ products are neither designed nor intended for use in military or aerospace applications or environments and for automotive applications or environment. Customer acknowledges and agrees that any such use of Arduino™ products which is solely at the Customer's risk, and that Customer is solely responsible for compliance with all legal and regulatory requirements in connection with such use.

1.6 Customer acknowledges and agrees that it is solely responsible for compliance with all legal, regulatory and safety-related requirements concerning its products and any use of Arduino™ products in Customer's applications, notwithstanding any applications-related information or support that may be provided by the producer.

2. Indemnification

The Customer acknowledges and agrees to defend, indemnify and hold harmless the producer from and against any and all third-party losses, damages, liabilities and expenses it incurs to the extent directly caused by: (i) an actual breach by a Customer of the representation and warranties made under this terms and conditions or (ii) the gross negligence or willful misconduct by the Customer.

3. Consequential Damages Waiver

In no event the producer shall be liable to the Customer or any third parties for any special, collateral, indirect, punitive, incidental, consequential or exemplary damages in connection with or arising out of the products provided hereunder, regardless of whether the producer has been advised of the possibility of such damages. This section will survive the termination of the warranty period.

4. Changes to specifications

The producer may make changes to specifications and product descriptions at any time, without notice. The Customer must not rely on the absence or characteristics of any features or instructions marked "reserved" or "undefined." The producer reserves these for future definition and shall have no responsibility whatsoever for conflicts or incompatibilities arising from future changes to them. The product information on the Web Site or Materials is subject to change without notice. Do not finalize a design with this information.

Enviromental Policies

The producer of Arduino™ has joined the Impatto Zero® policy of LifeGate.it. For each Arduino board produced is created / looked after half squared Km of Costa Rica's forest's.

Grove - Gas Sensor (MQ2) User Manual

Release date： 2015/9/22

Version： 1.0

Wiki: http://seeedstudio.com/wiki/Twig_-_Gas_Sensor%28MQ2%29

Bazaar: http://www.seeedstudio.com/depot/Grove-Gas-SensorMQ2-p-937.html

Document Revision History

Revision	Date	Author	Description
1.0	Sep 22, 2015	Jiankai.li	Create file

Contents

Disclaimer

For physical injuries and possessions loss caused by those reasons which are not related to product quality, such as operating without following manual guide, natural disasters or force majeure, we take no responsibility for that.

Under the supervision of Seeed Technology Inc., this manual has been compiled and published which covered the latest product description and specification. The content of this manual is subject to change without notice.

Copyright

1. Introduction

The Grove - Gas Sensor(MQ2) module is useful for gas leakage detecting(in home and industry). It can detect H2, LPG, CH4, CO, Alcohol, Smoke, Propane. Based on its fast response time. Measurements can be taken as soon as possible. Also the sensitivity can be adjusted by the potentiometer.

2. Features

- Wide detecting scope
- Stable and long life
- Fast response and High sensitivity

3. Application Ideas

- Gas leakage detecting
- Toys

4. Mechanic Dimensions

4.1 Electronic Characteristics

Items	Parameter name	Min	Type	Max	Unit
System Characteristics					
VCC	Working Voltage	4.9	5	5.1	V
PH	Heating consumption	0.5	-	800	mW
RL	Load resistance		can adjust		
RH	Heater resistance	-	33	-	Ω
Rs	Sensing Resistance	3	-	30	kΩ

5. Usage

5.1 Suggest Reading for Starter

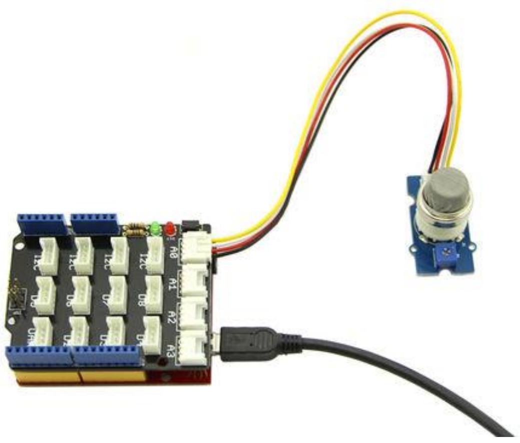
- Download Arduino and install Arduino driver
- Getting Started with Seeeduino
- How to choose a Gas Sensor
- What's LEL

5.2 Hardware Installation

Grove products have a eco system and all have a same connector which can plug onto the Base Shield. Connect this module to the A0 port of Base Shield, however, you can also connect Gas sensor to Arduino without Base Shield by jumper wires.

Arduino UNO	Gas Sensor
5V	VCC
GND	GND
NC	NC
Analog A0	SIG

You can gain the present voltage through the SIG pin of sensor. The higher the concentration of the gas, the bigger the output voltage of the SIG pin. Sensitivity can be regulated by rotating the potentiometer. Please note the best preheat time of the sensor is above 24 hours. For the detailed information about the MQ-2 sensor please refer to the datasheet.



5.3 How to use

There're two steps you need to do before getting the concentration of gas.

First, connect the module with Grove Shield using A0 like the picture above. And put the sensor in a clear air and use the program below.

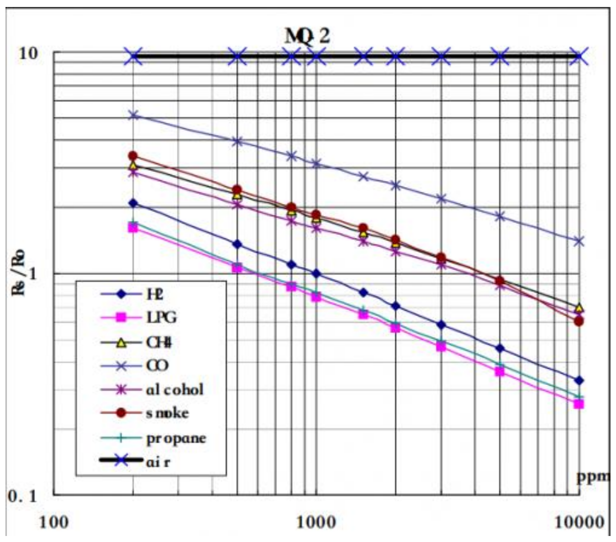
```
void setup() {
  Serial.begin(9600);
}

void loop() {
  float sensor_volt;
  float RS_air; //  Get the value of RS via in a clear air
  float R0;  // Get the value of R0 via in H2
  float sensorValue;

/*--- Get a average data by testing 100 times ---*/
    for(int x = 0 ; x < 100 ; x++)
  {
    sensorValue = sensorValue + analogRead(A0);
  }
  sensorValue = sensorValue/100.0;
/*-----------------------------------------------*/

  sensor_volt = sensorValue/1024*5.0;
  RS_air = (5.0-sensor_volt)/sensor_volt; // omit *RL
  R0 = RS_air/10.0; // The ratio of RS/R0 is 10 in a clear air

  Serial.print("sensor_volt = ");
  Serial.print(sensor_volt);
  Serial.println("V");

  Serial.print("R0 = ");
  Serial.println(R0);
  delay(1000);

}
```

Then, open the monitor of Arduino IDE, you can see some data are printed, write down the value of R0 and you need to use it in the following program. During this step, you may pay a while time to test the value of R0.

Second, put the sensor in one gas where the environment you want to test in. However, don't forget to replace the R0 below with value of R0 tested above.

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

void loop() {

  float sensor_volt;
  float RS_gas; // Get value of RS in a GAS
  float ratio; // Get ratio RS_GAS/RS_air
  int sensorValue = analogRead(A0);
  sensor_volt=(float)sensorValue/1024*5.0;
  RS_gas = (5.0-sensor_volt)/sensor_volt; // omit *RL

  /*-Replace the name "R0" with the value of R0 in the demo of First Test -*/
  ratio = RS_gas/R0;  // ratio = RS/R0
  /*-----------------------------------------------------------------------*/

  Serial.print("sensor_volt = ");
  Serial.println(sensor_volt);
  Serial.print("RS_ratio = ");
  Serial.println(RS_gas);
  Serial.print("Rs/R0 = ");
  Serial.println(ratio);

  Serial.print("\n\n");

  delay(1000);

}
```

Now, we can get the concentration of gas from the below figure



According to the figure, we can see that the minimum concentration we can test is 100ppm and the maximum is 10000ppm, in a other word, we can get a concentration of gas between 0.01% and 1%. However, we can't provide a formula because the relation between ratio and concentration is nonlinear.

6. Version Tracker

Revision	Descriptions	Release
v0.9b	Initial public release	16,Aug,2011
v1.4	Replace some components	27,Aug,2014

7. Resources

- File:Gas Sensor Eagle files.zip
- File:Gas Sensor Schematic.pdf
- File:MQ-2.pdf

8. Related Projects

If you want to make some awesome projects by Gas Sensor(MQ2) , here's some projects for reference.

8.1 Gas Sensor Demo

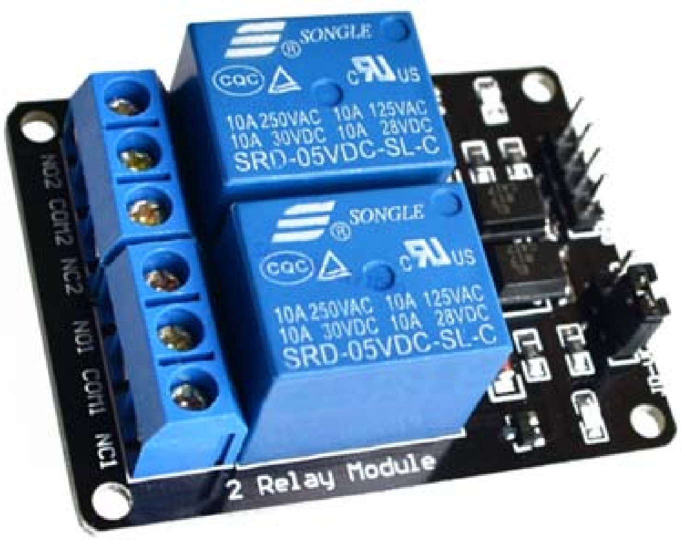
This is a demo about Air Quality Box make by Gas Sensor.

I want to make it.

User Guide

2 Channel 5V Optical Isolated Relay Module

This is a LOW Level 5V 2-channel relay interface board, and each channel needs a 15-20mA driver current. It can be used to control various appliances and equipment with large current. It is equipped with high-current relays that work under AC250V 10A or DC30V 10A. It has a standard interface that can be controlled directly by microcontroller. This module is optically isolated from high voltage side for safety requirement and also prevent ground loop when interface to microcontroller.



Brief Data:

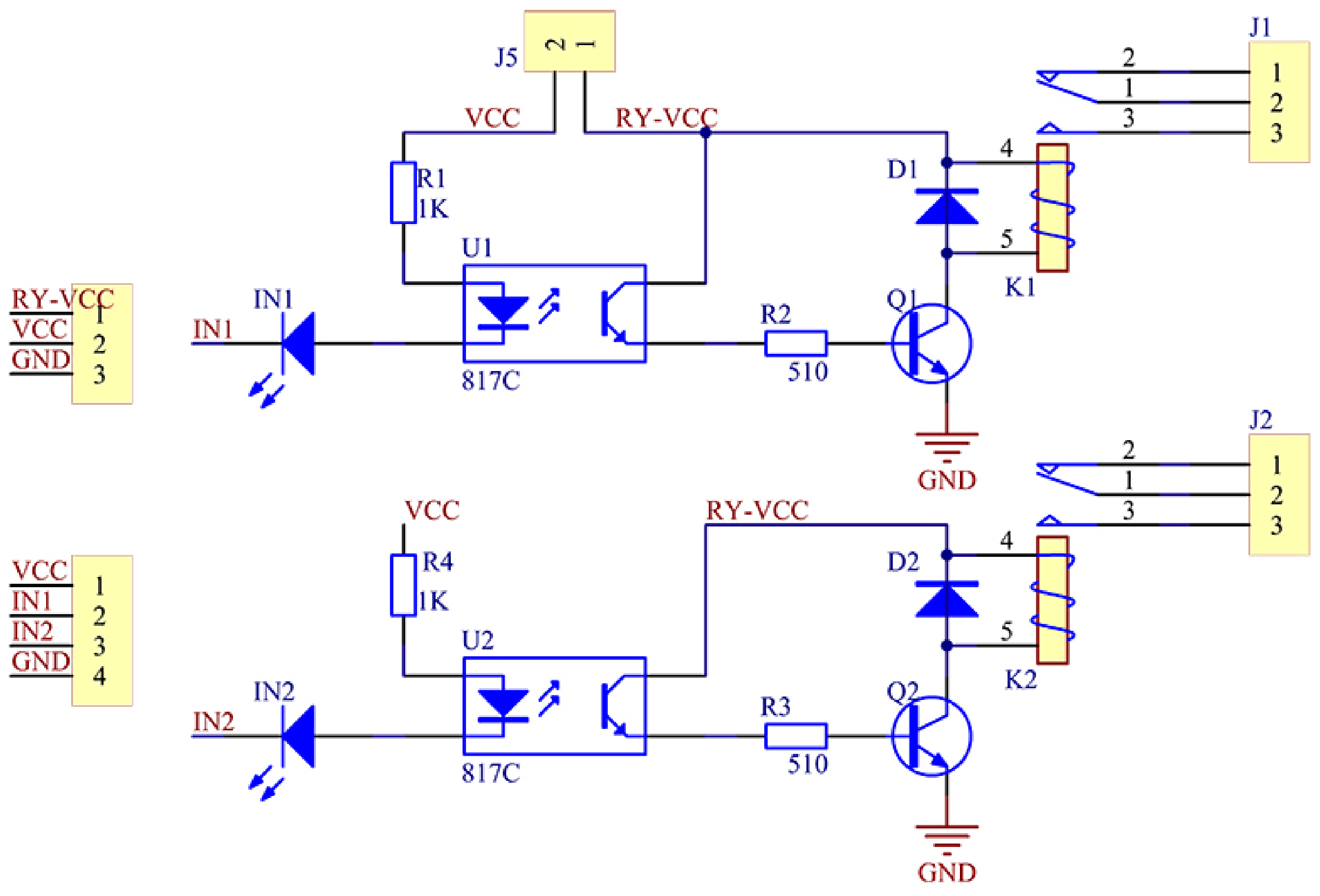
- Relay Maximum output: DC 30V/10A, AC 250V/10A.
- 2 Channel Relay Module with Opto-coupler. LOW Level Trigger expansion board, which is compatible with Arduino control board.
- Standard interface that can be controlled directly by microcontroller (8051, AVR, *PIC, DSP, ARM, ARM, MSP430, TTL logic).
- Relay of high quality low noise relays SPDT. A common terminal, a normally open, one normally closed terminal.
- Opto-Coupler isolation, for high voltage safety and prevent ground loop with microcontroller.

Schematic:

VCC and RY-VCC are also the power supply of the relay module. When you need to drive a large power load, you can take the jumper cap off and connect an extra power to RY-VCC to supply the relay; connect VCC to 5V of the MCU board to supply input signals.

NOTES: If you want complete optical isolation, connect "Vcc" to Arduino +5 volts but do NOT connect Arduino Ground. Remove the Vcc to JD-Vcc jumper. Connect a separate +5 supply to "JD-Vcc" and board Gnd. This will supply power to the transistor drivers and relay coils.

If relay isolation is enough for your application, connect Arduino +5 and Gnd, and leave Vcc to JD-Vcc jumper in place.

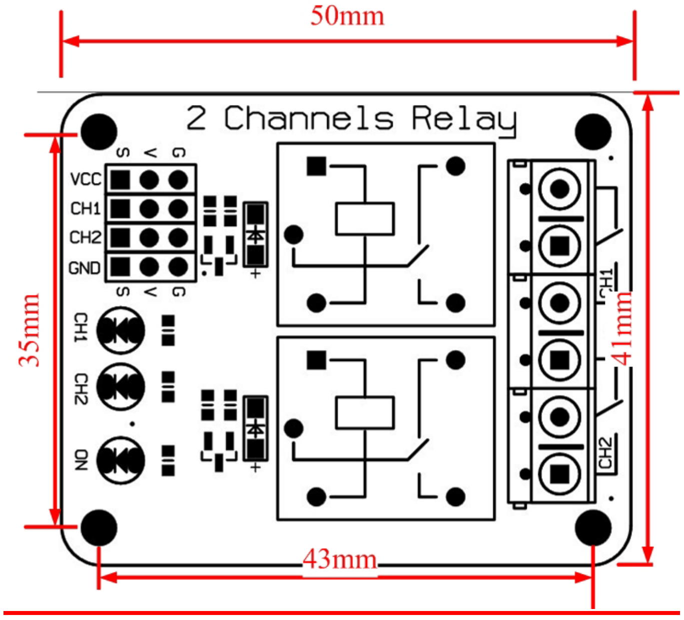


It is sometimes possible to use this relay boards with 3.3V signals, if the JD-VCC (Relay Power) is provided from a +5V supply and the VCC to JD-VCC jumper is removed. That 5V relay supply could be totally isolated from the 3.3V device, or have a common ground if opto-isolation is not needed. If used with isolated 3.3V signals, VCC (To the input of the opto-isolator, next to the IN pins) should be connected to the 3.3V device's +3.3V supply.

NOTE: Some Raspberry-Pi users have found that some relays are reliable and others do not actuate sometimes. It may be necessary to change the value of R1 from 1000 ohms to something like 220 ohms, or supply +5V to the VCC connection.

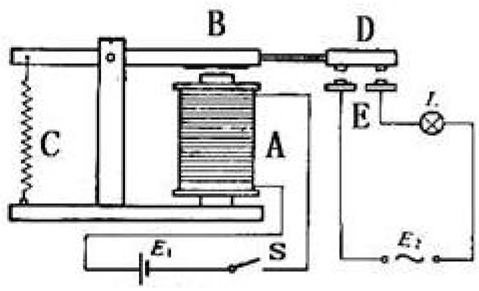
NOTE: The digital inputs from Arduino are Active LOW: The relay actuates and LED lights when the input pin is LOW, and turns off on HIGH.

Module Layout:



Operating Principle:

See the picture below: A is an electromagnet, B armature, C spring, D moving contact, and E fixed contacts. There are two fixed contacts, a normally closed one and a normally open one. When the coil is not energized, the normally open contact is the one that is off, while the normally closed one is the other that is on.



Supply voltage to the coil and some currents will pass through the coil thus generating the electromagnetic effect. So the armature overcomes the tension of the spring and is attracted to the core, thus closing the moving contact of the armature and the normally open (NO) contact or you may say releasing the former and the normally closed (NC) contact. After the coil is de-energized, the electromagnetic force disappears and the armature moves back to the original position, releasing the moving contact and normally closed contact. The closing and releasing of the contacts results in power on and off of the circuit.

Input:

VCC : Connected to positive supply voltage (supply power according to relay voltage)

GND : Connected to supply ground.

IN1: Signal triggering terminal 1 of relay module

IN2: Signal triggering terminal 2 of relay module

Output:

Each module of the relay has one NC (normally close), one NO (normally open) and one COM (Common) terminal. So there are 2 NC, 2 NO and 2 COM of the channel relay in total. NC stands for the normal close port contact and the state without power. NO stands for the normal open port contact and the state with power. COM means the common port. You can choose NC port or NO port according to whether power or not.

Testing Setup:

When a low level is supplied to signal terminal of the 2-channel relay, the LED at the output terminal will light up. Otherwise, it will turn off. If a periodic high and low level is supplied to the signal terminal, you can see the LED will cycle between on and off.

For Arduino：

Step 1：

Connect the signal terminal IN1、IN2 of 2-channel relay to digital pin 4 & 5 of the Arduino Uno or ATMega2560 board, and connect an LED at the output terminal.

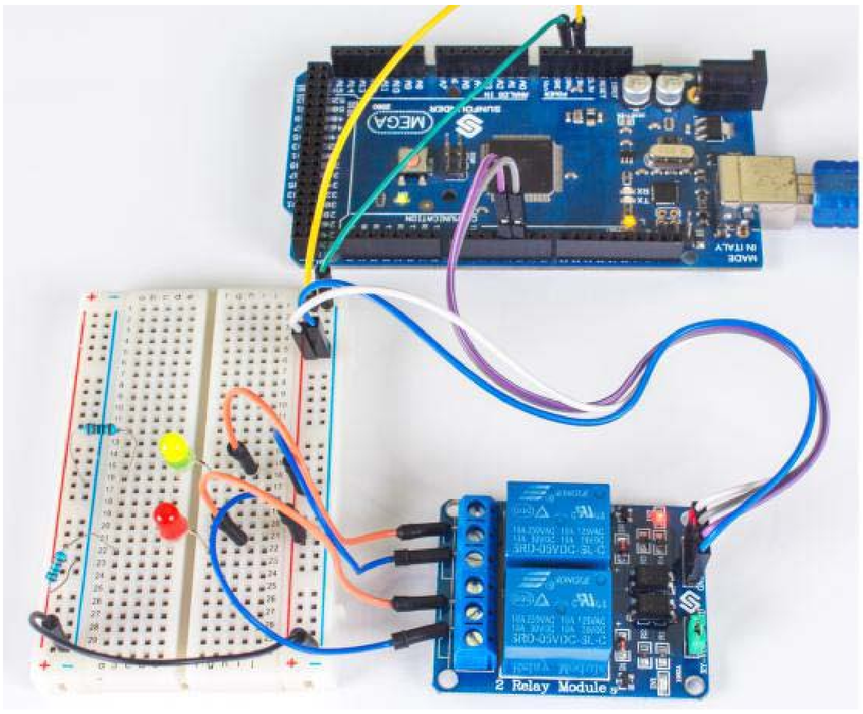
IN1> 4

IN2> 5

Step 2:

Upload the sketch "text_code" to the Arduino Uno or ATMega2560 board.Then you can see the LED cycle between on and off.

The actual figure is shown below:



For raspberry Pi:

Step1:

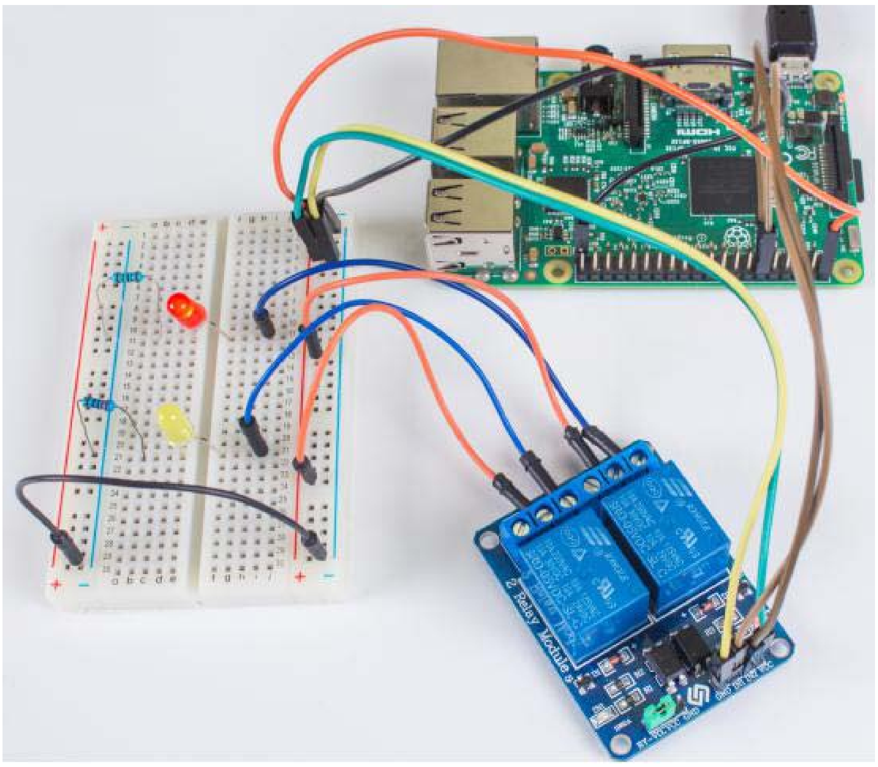
Connect the signal terminal IN2、IN1 of 2-channel relay to port 17、18 of the Raspberry Pi, and connect an LED at the output terminal.

IN2 > 17

IN1 > 18

Step 2:

Run the “test_code”. Then you can see the LED cycle between on and off.



Sketch for Arduino:

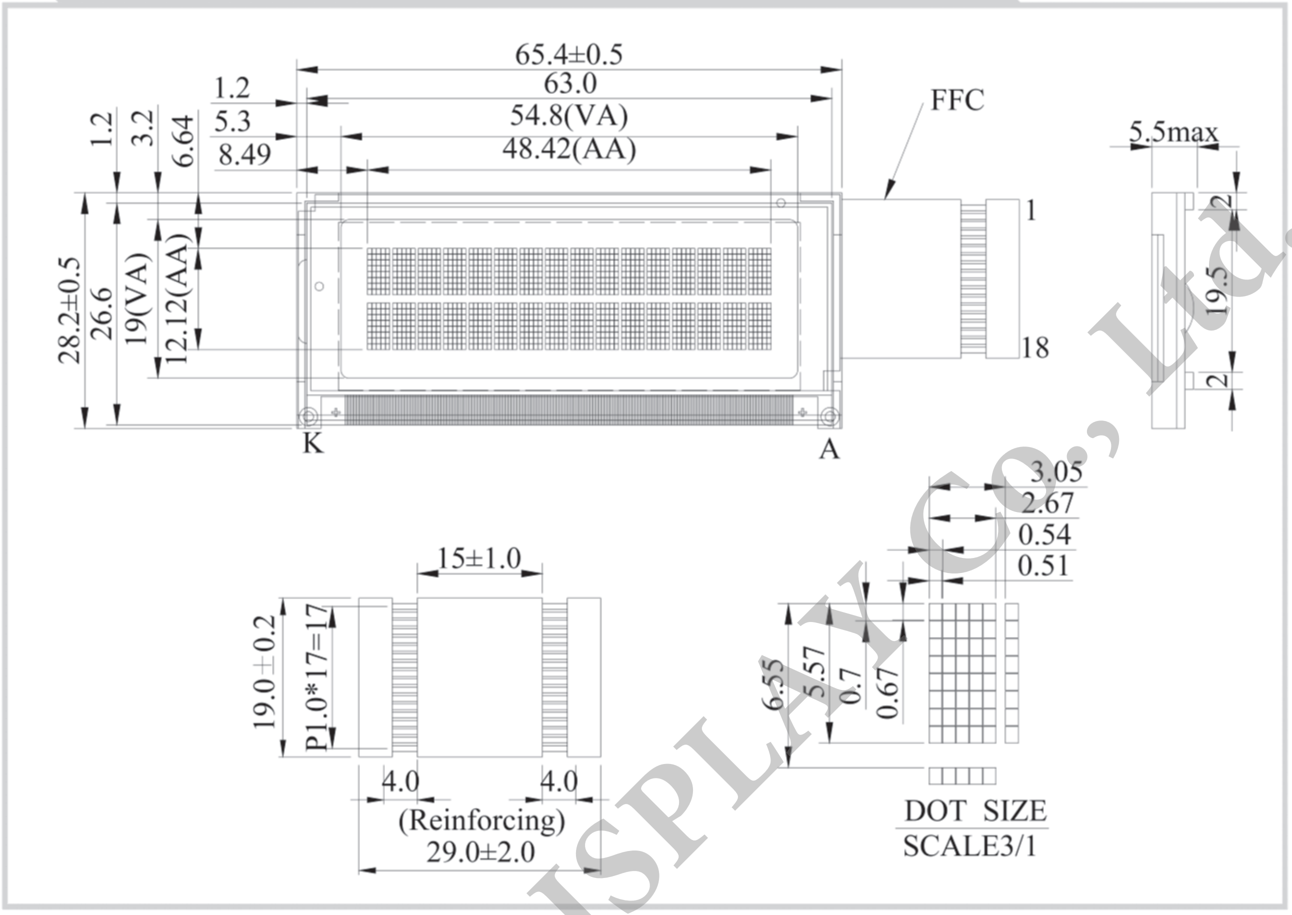
```
/**************************************************
   Name：_2_channel_relay
   Description: control the 2 channel relay module to ON or OFF
   Website: www.handsontec.com
   Email: techsupport@handsontec.com
******************************************************/

//the relays connect to
int IN1 = 4;
int IN2 = 5;

#define ON   0
#define OFF  1


void setup()
{
  relay_init();//initialize the relay
}

void loop() {
  relay_SetStatus(ON, OFF);//turn on RELAY_1
  delay(2000);//delay 2s
  relay_SetStatus(OFF, ON);//turn on RELAY_2
  delay(2000);//delay 2s
}
void relay_init(void)//initialize the relay
{
  //set all the relays OUTPUT
  pinMode(IN1, OUTPUT);
  pinMode(IN2, OUTPUT);
  relay_SetStatus(OFF, OFF); //turn off all the relay
}
//set the status of relays
void relay_SetStatus( unsigned char status_1,  unsigned char status_2)
{
  digitalWrite(IN1, status_1);
  digitalWrite(IN2, status_2);
}
```

Code for Raspberry Pi:

```
#!/usr/bin/env python
'''
**********************************************************************
* Filename      : 2_channel_relay.py
* Description   : a sample script for 2-Channel High trigger Relay
* E-mail        : techsupport@handsontec.com
* Website       : www.handsontec.com
* Detail        : New file
**********************************************************************
'''
import RPi.GPIO as GPIO
from time import sleep

Relay_channel = [17, 18]

def setup():
    GPIO.setmode(GPIO.BOARD)
    GPIO.setup(Relay_channel, GPIO.OUT, initial=GPIO.LOW)
    print "|=====================================================|"
    print "|         2-Channel High trigger Relay Sample         |"
    print "|-----------------------------------------------------|"
    print "|                                                     |"
    print "|          Turn 2 channels on off in orders           |"
    print "|                                                     |"
    print "|                    17 ===> IN2                      |"
    print "|                    18 ===> IN1                      |"
    print "|                                                     |"
    print "|                                                     |"
    print "|=====================================================|"

def main():
    while True:
        for i in range(0, len(Relay_channel)):
            print '...Relay channel %d on' % i+1
            GPIO.output(Relay_channel[i], GPIO.HIGH)
            sleep(0.5)
            print '...Relay channel %d off' % i+1
            GPIO.output(Relay_channel[i], GPIO.LOW)
            sleep(0.5)

def destroy():
    GPIO.output(Relay_channel, GPIO.LOW)
    GPIO.cleanup()
```

```
if __name__ == '__main__':
    setup()
    try:
        main()
    except KeyboardInterrupt:
        destroy()
```

WH1602T Character 16x2



Feature

1.5x8 dots includes cursor
2.Bulit-in controller (ST7066 or Equivalent)
3.5V power supply only
4.N.V, optional for 3V power supply
5.1/16 duty cycle
6.White LED B/L not available
7.Interface : 6800, option SPI/I2C (RW1063 IC)

Pin No.	Symbol	Description
1	V_{DD}	Power supply for logic
2	V_{SS}	Ground
3	V_O	Contrast Adjustment
4	NC	No connection
5	NC	No connection
6	RS	Data/ Instruction select signal
7	R/W	Read/Write select signal
8	E	Enable signal
9	DB0	Data bus line
10	DB1	Data bus line
11	DB2	Data bus line
12	DB3	Data bus line
13	DB4	Data bus line
14	DB5	Data bus line
15	DB6	Data bus line
16	DB7	Data bus line
17	A	Power supply for B/L +
18	K	Power supply for B/L -

Mechanical Data

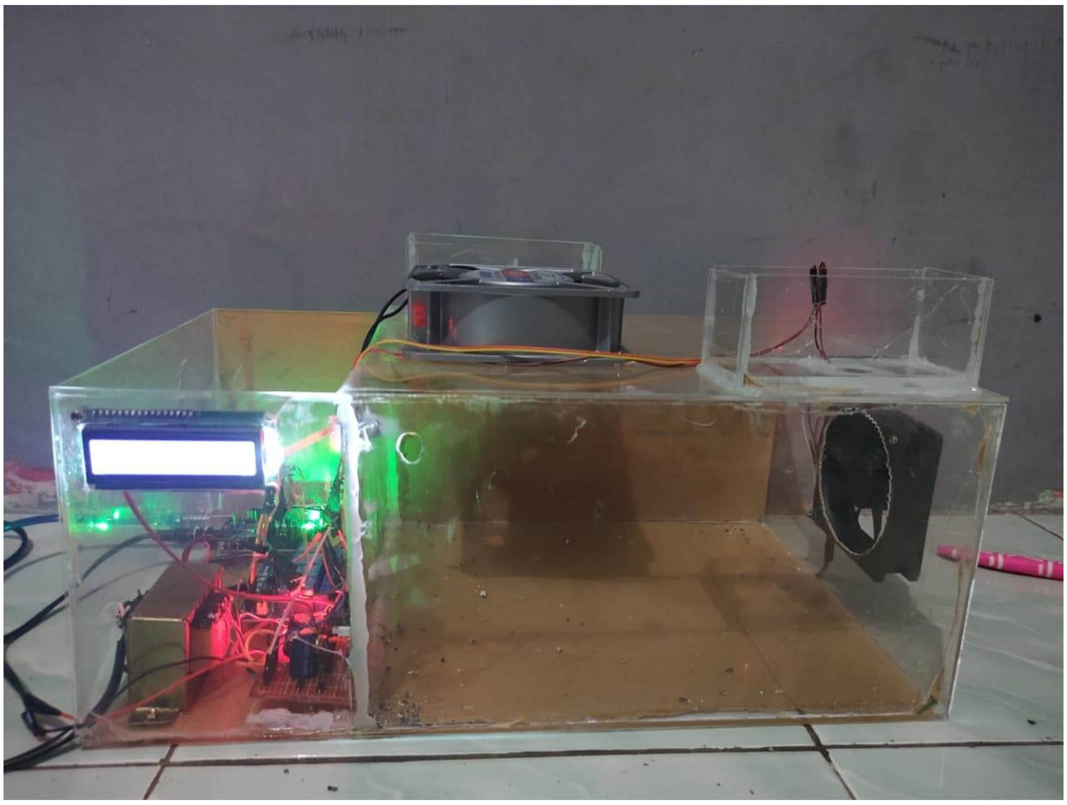
Item	Standard Value	Unit
Module Dimension	65.4 x 28.2	mm
Viewing Area	54.8 x 19.0	mm
Character Size	2.67 x 5.57	mm

Electrical Characteristics

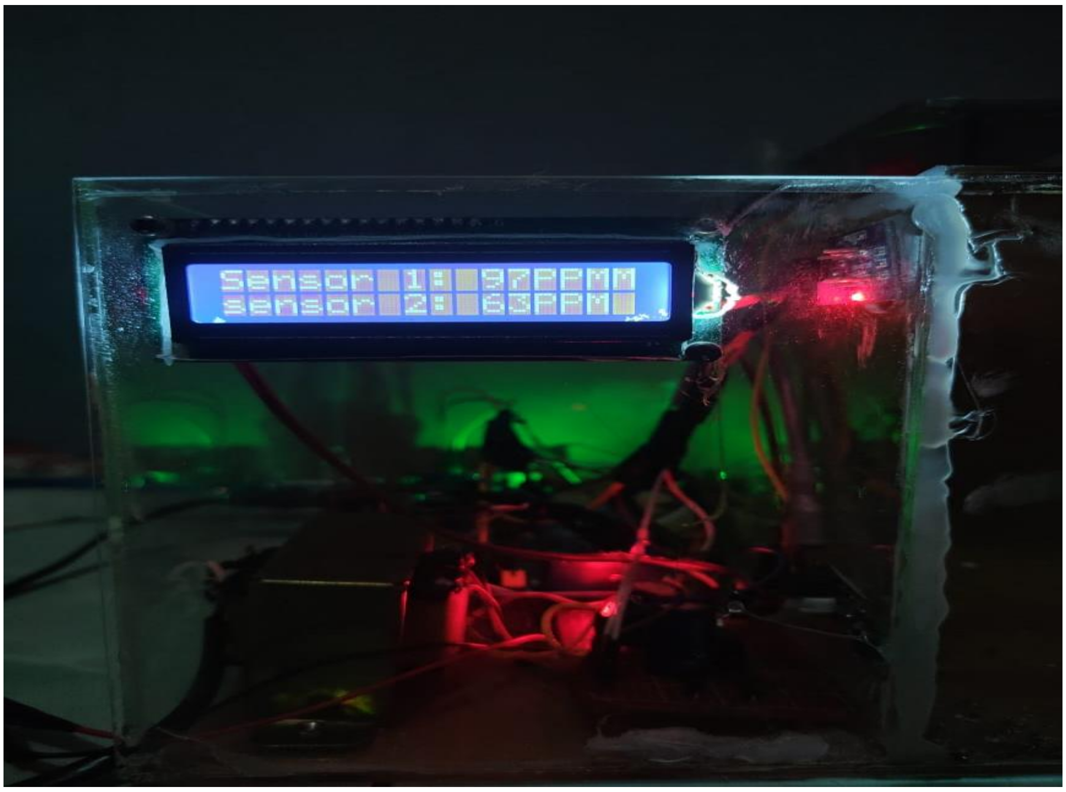
Item	Symbol	Standard Value	Unit
		typ.	
Input Voltage	VDD	5.0	V
Recommended LCD Driving Voltage for Normal Temp. Version module @25°C	VDD-VO	4.20	V

Display Character Address Code

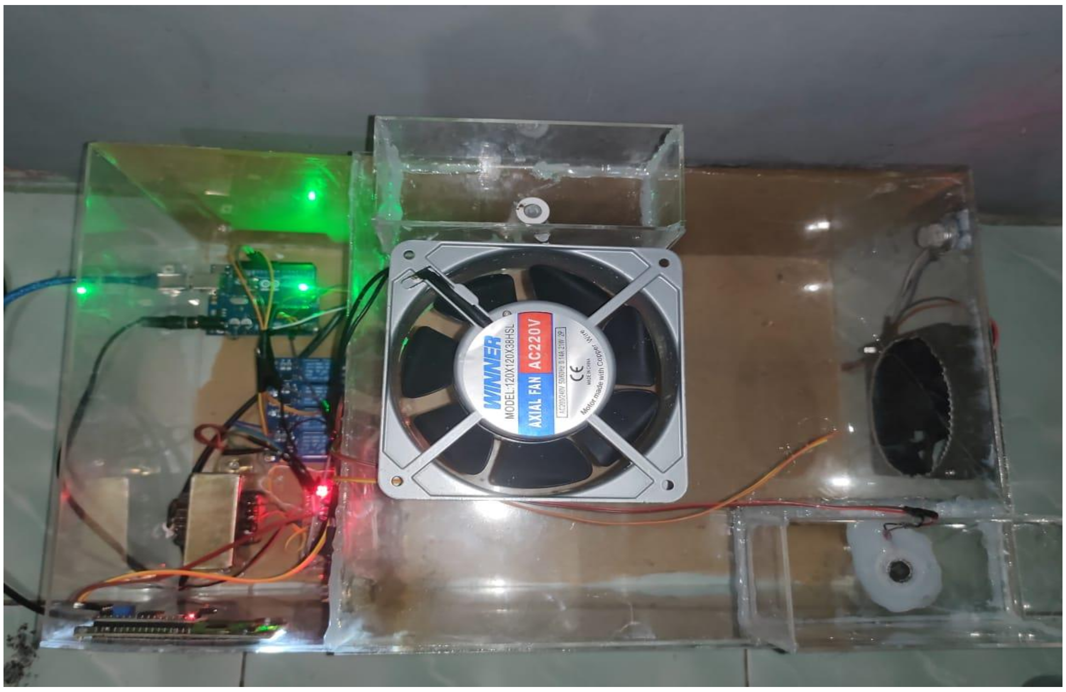
Display position	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
DD RAM Address	00	01	02												0E	0F
DD RAM Address	40	41	42												4E	4F



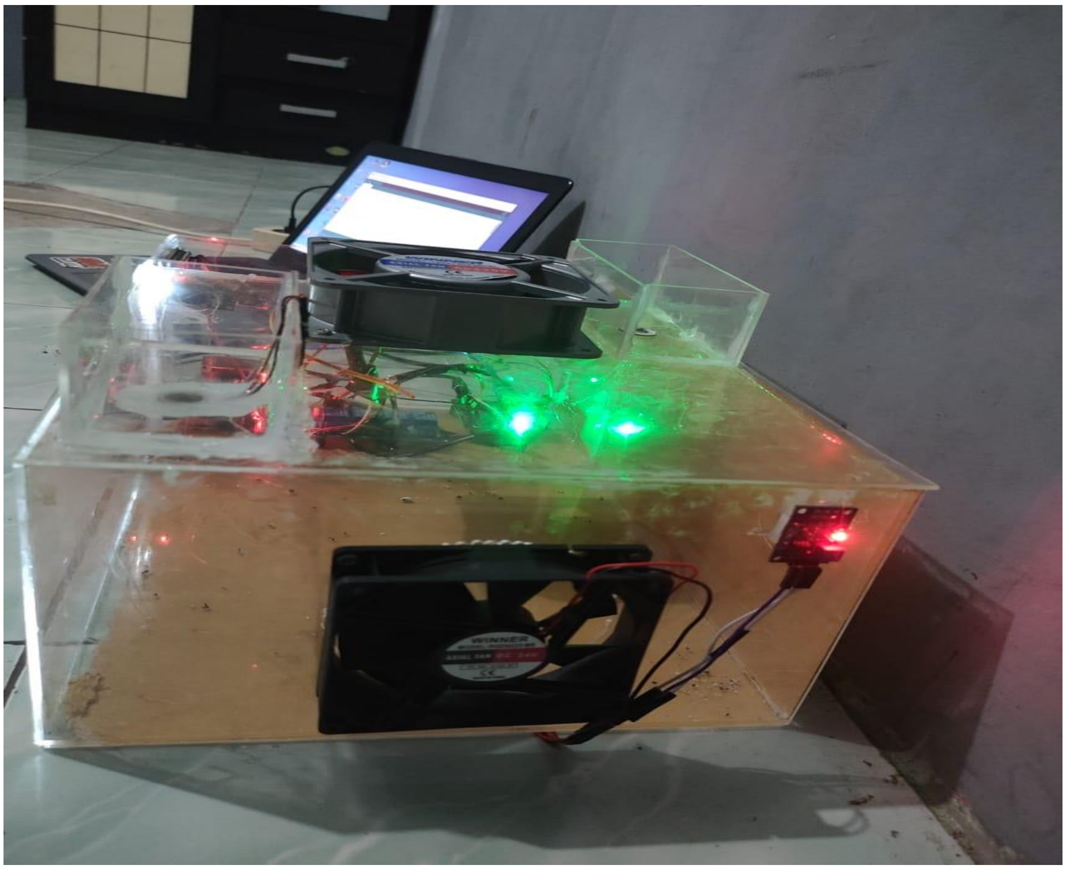
Gambar. 1 Tampilan Keseluruhan Alat



Gambar. 2 Bentuk Fisik LCD



Gambar. 3 Bentuk Fisik KIPAS 1



Gambar. 4 Bentuk Fisik KIPAS 2