KEMENTERIAN RISET, TEKNOLOGI DAN PENDIDIKAN TINGGI
POLITEKNIK NEGERI SRIWIJAYA
JalanSrijaya Negara, Palembang 30139
Telp. 0711-353414 Fax. 0711-355918
Website : www.polisriwijaya.ac.id **E-mail :** info@polsri.ac.id

KESEPAKATAN BIMBINGAN LAPORAN AKHIR (LA)

Kami yang bertanda tangan di bawahini,

PihakPertama

Nama Mahasiswa	: Diah Anggraini
NIM	: 0614 3032 0196
Jurusan	: Teknik Elektro
Program Studi	: Teknik Elektronika

Pihak Kedua

Nama	: Ir. A. Rahman, M.T
NIP	: 196202051993031002
Jurusan	: Teknik Elektro
Program Studi	: Teknik Elektronika

Pada hari ini Senin.... tanggal 6 Maret ..... telah sepakat untuk melakukan konsultasi bimbingan Laporan Akhir.
Konsultasi bimbingan sekurang-kurangnya 1 (satu) kali dalam satu minggu. Pelaksanaan bimbingan pada setiap hari ..... Pukul ..... tempat di Politeknik Negeri Sriwijaya.
Demikianlah kesepakatan ini dibuat dengan penuh kesadaran guna kelancaran penyelesaian Laporan Akhir.

Palembang, 6 Maret 2017

PihakPertama,

Diah Anggraini
NIM. 0614 3032 0196

PihakKedua,

Ir. A. Rahman, M.T
NIP. 196202051993031002

Mengetahui,
KetuaProgram Studi

Amperawan, ST., M.T.
NIP.196705231993031002

KEMENTERIAN RISET, TEKNOLOGI DAN PENDIDIKAN TINGGI
POLITEKNIK NEGERI SRIWIJAYA
JalanSrijaya Negara, Palembang 30139
Telp. 0711-353414 Fax. 0711-355918
Website : www.polisriwijaya.ac.id **E-mail :** info@polsri.ac.id

KESEPAKATAN BIMBINGAN LAPORAN AKHIR (LA)

Kami yang bertandatangan di bawahini,

Pihak Pertama

Nama Mahasiswa	: Diah Anggraini
NIM	: 0614 3032 0196
Jurusan	: Teknik Elektro
Program Studi	: Teknik Elektronika

Pihak Kedua

Nama	: Niksen Alfarizal,S.T.,M.Kom
NIP	: 197508162001121001
Jurusan	: Teknik Elektro
Program Studi	: Teknik Elektronika

Pada hari ini Senin tanggal 6/3/17 telah sepakat untuk melakukan konsultasi bimbingan Laporan Akhir.
Konsultasi bimbingan sekurang-kurangnya 1 (satu) kali dalam satu minggu. Pelaksanaan bimbingan pada setiap hari Senin Pukul 09.00 tempat di Politeknik Negeri Sriwijaya.
Demikianlah kesepakatan ini dibuat dengan penuh kesadaran guna kelancaran penyelesaian Laporan Akhir.

Palembang, 6 - maret - 2017

PihakPertama,

Diah Anggraini
NIM. 061430321138

PihakKedua,

Niksen Alfarizal, S.T.,M.Kom
NIP. 197508162001121001

Mengetahui,
KetuaProgram Studi

Amperawan, ST., M.T.
NIP.196705231993031002

KEMENTERIAN RISET, TEKNOLOGI DAN PENDIDIKAN TINGGI
POLITEKNIK NEGERI SRIWIJAYA
Jalan Srijaya Negara, Palembang 30139
Telp. 0711-353414 Fax. 0711-355918
Website : www.polisriwijaya.ac.id E-mail : info@polsri.ac.id

LEMBAR BIMBINGAN LAPORAN AKHIR

Lembar : 1

Nama Mahasiswa : **Diah Anggraini**

NIM : **0614 3032 0196**

Jurusan/ Program Studi : **Teknik Elektro / Teknik Elektronika**

Judul Laporan Akhir : **Aplikasi Arduino Mega 2560 Dalam Rancang Bangun Alat Kontrol Kadar pH air pada Tambak Udang**

Dosen Pembimbing I : **Ir. A. Rahman, M.T**

NIP : **196202051993031002**

No	Tanggal	Uraian Bimbingan	Tanda Tangan Pembimbing
1.	6/3-2017	Seminar bimbingan	
2.	9/3-2017	Perbaikan Bab I	
3.	27/3-2017	Out line akhir bab. I .	
4.	3/4-2017	Bab. II .	
5.	10/4-2017	Konsultasi lap. bab. II	
6.	17/4-2017	Perb. Bab III	
7.	2/5-2017	Konsultasi Lap. hasil Bab. IV	

No	Tanggal	Uraian Bimbingan	Tanda Tangan Pembimbing
8.	3/5-2017	Konsultasi bab I, II, III	
9.	2/6-2017. 6	Konsultasi alat yang di Rancang.	
10.	6/6-2017	Pengisian Lap. bab. IV & V	
11.	6/7-2017.	Rekomendasi U/ujian Lap° akhir	
12.			
13.			
14.			

Palembang, ...

Ketua Program Studi,

Amperawan, ST., M.T.
NIP.196705231993031002

Catatan:
Ketua Jurusan/Ketua Program Studi harus memeriksa jumlah pelaksanaan bimbingan sesuai yang dipersyaratkan dalam Pedoman Laporan Akhir sebelum menandatangani Lembar bimbingan ini,
Lembar pembimbingan LA ini harus dilampirkan dalam Laporan Akhir

KEMENTERIAN RISET, TEKNOLOGI DAN PENDIDIKAN TINGGI
POLITEKNIK NEGERI SRIWIJAYA
Jalan Srijaya Negara, Palembang 30139
Telp. 0711-353414 Fax. 0711-355918
Website : www.polisriwijaya.ac.id E-mail : info@polsri.ac.id

LEMBAR BIMBINGAN LAPORAN AKHIR

Lembar : 1

Nama Mahasiswa : **Diah Anggraini**

NIM : **0614 3032 0196**

Jurusan/ Program Studi : **Teknik Elektro / Teknik Elektronika**

Judul Laporan Akhir : **Aplikasi Arduino Mega 2560 Dalam Rancang Bangun Alat Kontrol Kadar pH air pada Tambak Udang**

Dosen Pembimbing II : **Niksen Alfarizal, S.T.,M.Kom**

NIP : **197508162001121001**

No	Tanggal	Uraian Bimbingan	Tanda Tangan Pembimbing
1.	7/3.17	Konsultasi proposal L.Akhir Kesepakatan bimbingan	Vik
2.	14/3.17	Perbaikan proposal Buat perencana alat yg baik shg dpt dibuat tepat waktu	Vik
3.	21/3.17	Acc proposal konsultasi yg pbbj I	Vik
4.	28/3.17	konsultasi BAB I/II ke pembimbing I	Vik
5.	5/4/17	Uji alat tahap I : Perbaiki kinerja alat / Pelayari thg Solar sel	Vik
6.	11/4.17	Uji alat tahap II : Perbaiki kinerja alat	Vik
7.	18/4.17	konsultasi pembimbing I :	Vik

Lembar : 2

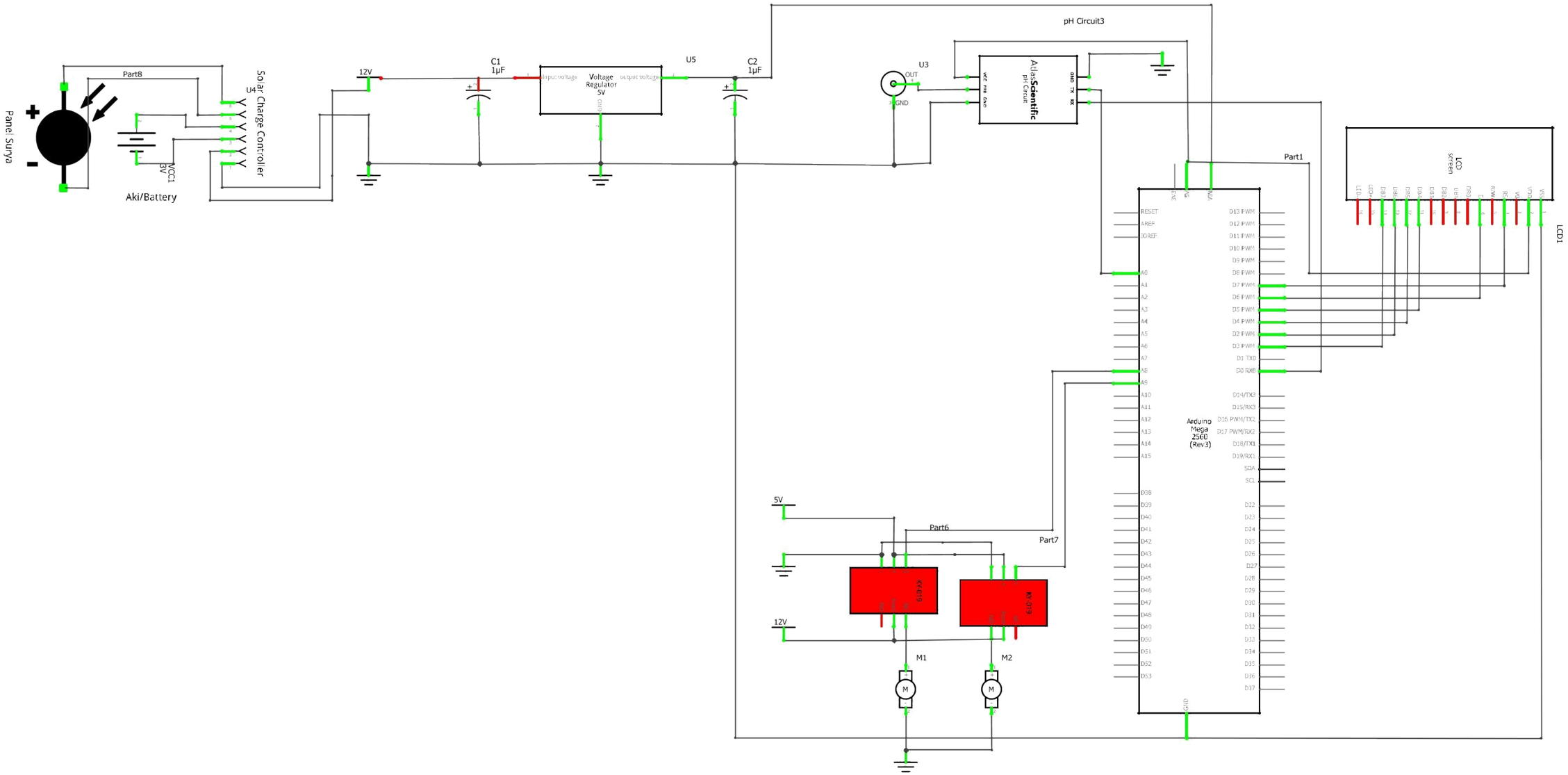
No	Tanggal	Uraian Bimbingan	Tanda Tangan Pembimbing
8.	25/4. 17	Perbaiki : konstruksi frame/ alat buat sesuai meja	Vik
9.	2/5. 17	perbaiki konsep alat/ konsultasi pembimbing I/	Vik
10.	10/5. 17	ACC alat tlah uji	Vik
11.	24/5 17	Lengkapi kelengkapan laporan daftar isi/tabel/pgbr-dll.	Vik
12.	6/7. 17.	ACC : Sidang / ujian L.A	Vik
13.			
14.			

Palembang, ...

Ketua Program Studi,

Amperawan, ST., M.T.
NIP.196705231993031002

Catatan:
Ketua Jurusan/Ketua Program Studi harus memeriksa jumlah pelaksanaan bimbingan sesuai yang dipersyaratkan dalam Pedoman Laporan Akhir sebelum menandatangani Lembar bimbingan ini,
Lembar pembimbingan LA ini harus dilampirkan dalam Laporan Akhir



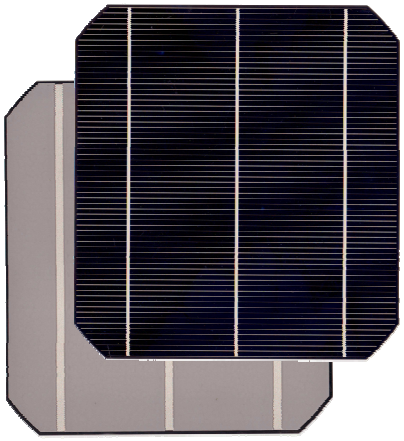
fritzing

3.11	RANGKAIAN SISTEM KESELURUHAN ALAT KONTROL KADAR PH AIR		
POLITEKNIK NEGERI SRIWIJAYA		NAMA : DIAH ANGGRAINI	HAL 46
		Tanggal : 28 Juni 2017	KELAS : 6EA

HIGH PERFORMANCE	DATASHEET	TG18.5 BR

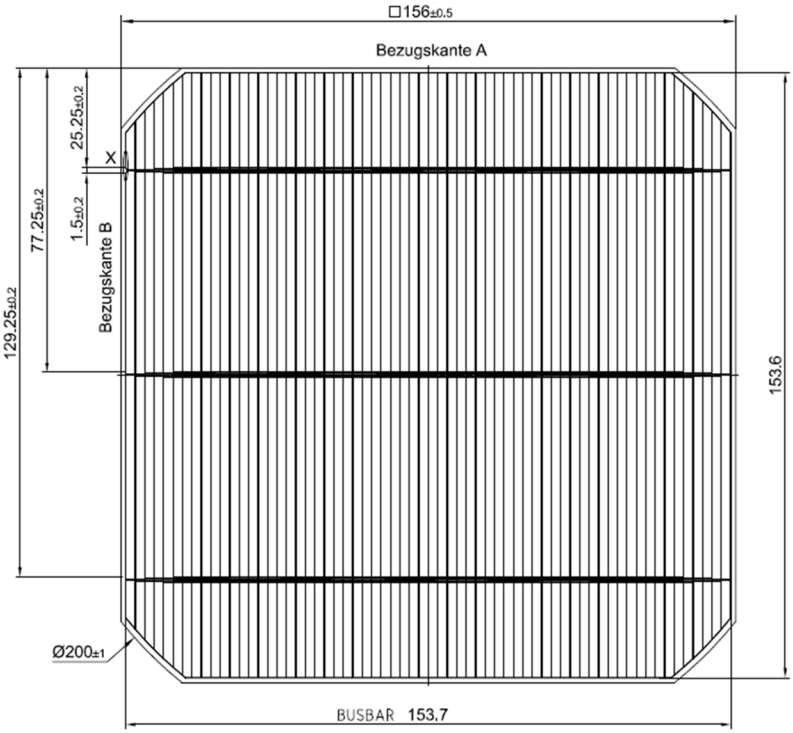
Monocrystalline Silicon Solar Cell

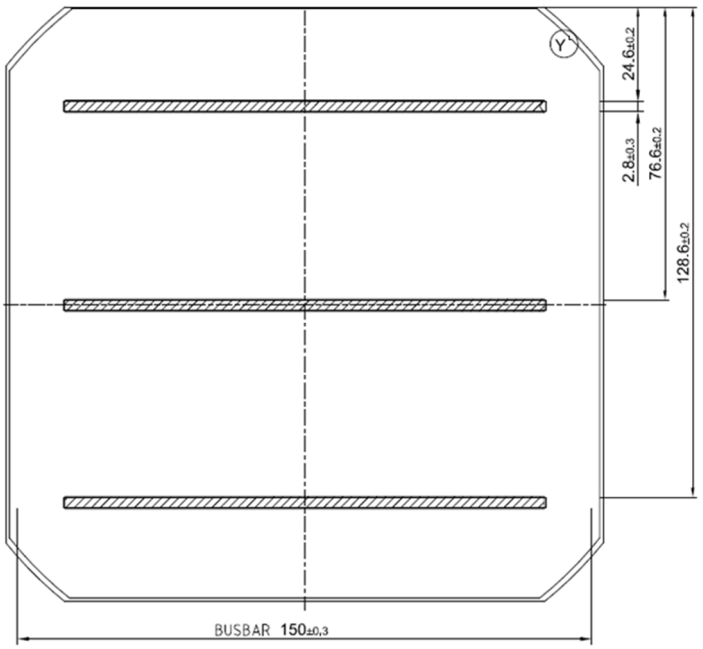
TG18.5 BR (D200, 156mm x 156mm)



OVERVIEW	
Product	Monocrystalline P-Type Silicon Solar Cell
Format ; Diameter	156 mm x 156 mm; 200 mm
Description:	High performance and premium optical quality and appearance cell, suitable for all applications including BIPV. (Efficiency 17.5% - 18.39%).
Electrical contacts	Front side: grid; 3 busbars
	Back side: 3 busbars

CELL LAYOUT





bSolar GmbH
○ Niederlassung Heilbronn, Theresienstraße 2, 74072 Heilbronn, (Tel)+49(0)7131-673353, (Fax)+49(0)7131-672233
bSolar Ltd.
○ 21 Havaad Haleumi st., Jerusalem 91160, Israel, (Phone/Fax): +972-74-7024797

HIGH PERFORMANCE	DATASHEET	TG18.5 BR	bSolar The Bifacial PV Cell Company

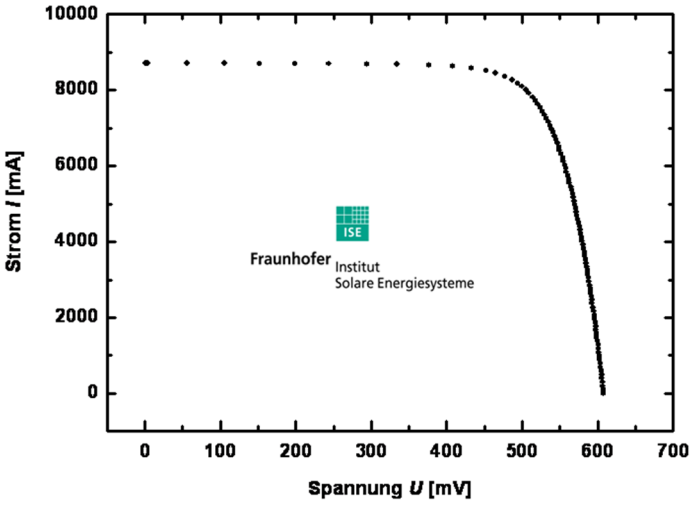
APPEARANCE AND DIMENSIONS	
Material	Monocrystalline Silicon
Surface and color	Textured, dark blue - black
Dimensions; area	156 mm x 156 mm; 23,895 mm² (±250 mm²)
Cell thickness (related to silicon)	180 µm (±30µm)
Electrical contacts	Front side: grid; 3 busbars
	Back side: 3 busbars
Polarity	Front side: negative; back side: positive
Cell structure	n+ p p+
Antireflective coating	Silicon Nitride

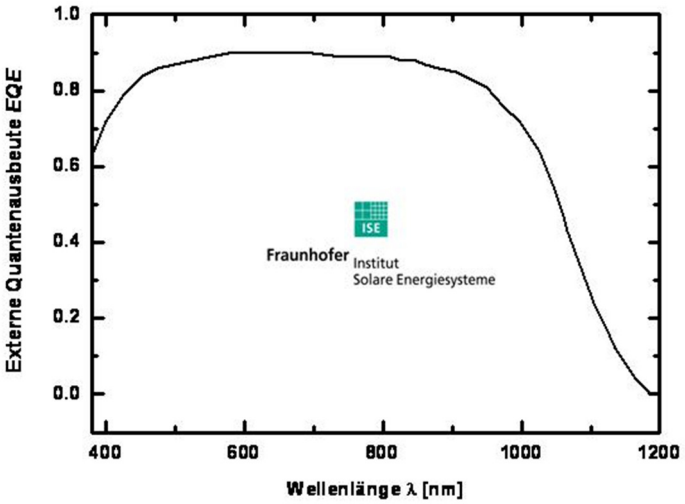
ELECTRICAL DATA*		BIN 36	BIN 35	BIN 34
Classification voltage	U_{LD} (mV)	500	500	500
Voltage at open circuit	U_{OC}(mV)	620	618	612
Current at classification voltage	I_{ULD} (A)	8.60-8.84	8.36-8.60	8.12-8.36
Mean short circuit current	I_{SC} (A)	8.88	8.85	8.75
Mean power	P_{MPP} (W)	4.39	4.35	4.18
Mean efficiency	η (%)	18.39	18.20	17.53
Reverse current	I_{Rev12V} (A)	≤ 0.5	≤ 0.5	≤ 0.5

TEMPERATURE COEFFICIENTS*		ABSOLUTE	RELATIVE
Voltage at open circuit	T_{KUOC}	-2.23 mV/K	-0.37 %/K
Short circuit current	T_{KISC}	3.14 mA/K	0.04 %/K
Power	T_{KP}	-18.02 mW/K	-0.45 %/K

* These values are valid for the following testing conditions: light spectrum AM1.5G; light intensity 100 mW/cm²; measuring temperature 25°C; accuracy in the range from 25°C to 75°C: ±2.5%

PRODUCT INFORMATION	
Working temperature of the cell	-50 °C to 80 °C
Stocking conditions	Avoid humidity and corrosive atmospheres
Recommended solder	Saturated with silver (2-4% Ag)





Specifications subject to technical changes

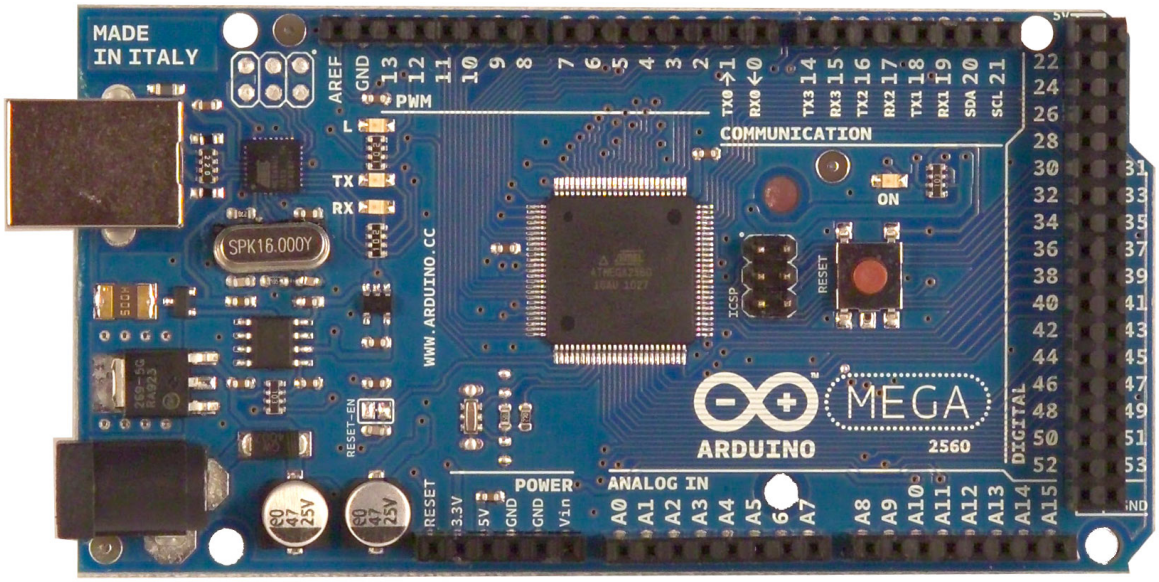
bSolar GmbH
○ Niederlassung Heilbronn, Theresienstraße 2, 74072 Heilbronn, (Tel)+49(0)7131-673353, (Fax)+49(0)7131-672233
bSolar Ltd.
○ 21 Havaad Haleumi st., Jerusalem 91160, Israel, (Phone/Fax): +972-74-7024797

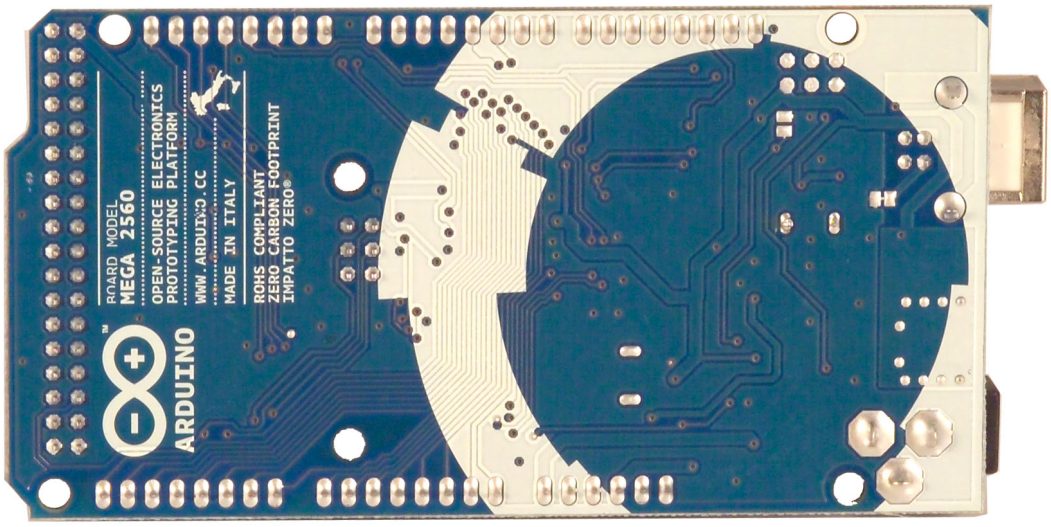
www.robotshop.com

La robotique à votre service! - Robotics at your service!

ARDUINO

Arduino Mega 2560 Datasheet





Overview

The Arduino Mega 2560 is a microcontroller board based on the ATmega2560 (datasheet). It has 54 digital input/output pins (of which 14 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), 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 Mega is compatible with most shields designed for the Arduino Duemilanove or Diecimila.

Schematic & Reference Design

EAGLE files: arduino-mega2560-reference-design.zip

Schematic: [arduino-mega2560-schematic.pdf]

Summary

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	54 (of which 14 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz

Power

The Arduino Mega 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 Mega2560 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.

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 ATmega2560 has 256 KB of flash memory for storing code (of which 8 KB is used for the bootloader), 8 KB of SRAM and 4 KB of EEPROM (which can be read and written with the EEPROM library).

Input and Output

Each of the 54 digital pins on the Mega 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); Serial 1: 19 (RX) and 18 (TX); Serial 2: 17 (RX) and 16 (TX); Serial 3: 15 (RX) and 14 (TX).** Used to receive (RX) and transmit (TX) TTL serial data. Pins 0 and 1 are also connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.
- **External Interrupts: 2 (interrupt 0), 3 (interrupt 1), 18 (interrupt 5), 19 (interrupt 4), 20 (interrupt 3), and 21 (interrupt 2).** 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: 0 to 13.** Provide 8-bit PWM output with the analogWrite() function.
- **SPI: 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS).** These pins support SPI communication using the SPI library. The SPI pins are also broken out on the ICSP header, which is physically compatible with the Uno, Duemilanove and Diecimila.
- **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.
- **I2C: 20 (SDA) and 21 (SCL).** Support I2C (TWI) communication using the Wire library (documentation on the Wiring website). Note that these pins are not in the same location as the I2C pins on the Duemilanove or Diecimila.

The Mega2560 has 16 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 analogReference() function.

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.

Communication

The Arduino Mega2560 has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega2560 provides four hardware UARTs for TTL (5V) serial communication. An ATmega8U2 on the board channels one of these over USB and provides a virtual com port to software on the computer (Windows machines will need a .inf file, but OSX and Linux machines will recognize the board as a COM port automatically. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the board. The RX and TX LEDs on the board will flash when data is being transmitted via the ATmega8U2 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 Mega2560's digital pins.
The ATmega2560 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the documentation on the Wiring website for details. For SPI communication, use the SPI library.

Programming

The Arduino Mega can be programmed with the Arduino software (download). For details, see the reference and tutorials.
The ATmega2560 on the Arduino Mega 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.

Automatic (Software) Reset

Rather then requiring a physical press of the reset button before an upload, the Arduino Mega2560 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 ATmega2560 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 Mega2560 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 Mega2560. 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 Mega2560 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 Mega2560 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 and Shield Compatibility

The maximum length and width of the Mega2560 PCB are 4 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.

The Mega2560 is designed to be compatible with most shields designed for the Uno, Diecimila or Duemilanove. Digital pins 0 to 13 (and the adjacent AREF and GND pins), analog inputs 0 to 5, the power header, and ICSP header are all in equivalent locations. Further the main UART (serial port) is located on the same pins (0 and 1), as are external interrupts 0 and 1 (pins 2 and 3 respectively). SPI is available through the ICSP header on both the Mega2560 and Duemilanove / Diecimila. *Please note that I^2C is not located on the same pins on the Mega (20 and 21) as the Duemilanove / Diecimila (analog inputs 4 and 5).*

PH meter(SKU: SEN0161)

Analog pH Meter Kit SKU: SEN0161

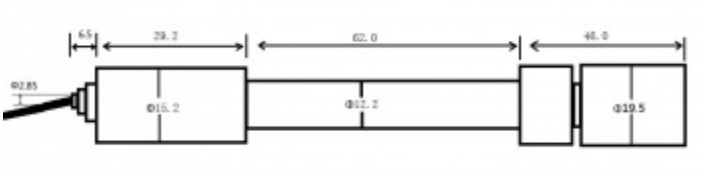
Analog pH Meter Kit SKU: SEN0169

Contents

Introduction

Need to measure water quality and other parameters but haven't got any low cost pH meter? Find it difficult to use with Arduino? Here comes an analog pH meter, specially designed for Arduino controllers and has built-in simple, convenient and practical connection and features. It has an LED which works as the Power Indicator, a BNC connector and PH2.0 sensor interface. You can just connect the pH sensor with BNC connector, and plug the PH2.0 interface into any analog input on Arduino controller to read pH value easily.

Specification



SEN0161 dimension

- Module Power: 5.00V
- Circuit Board Size: 43mm×32mm
- pH Measuring Range: 0-14
- Measuring Temperature: 0-60 ℃
- Accuracy: ± 0.1pH (25 ℃)
- Response Time: ≤ 1min
- pH Sensor with BNC Connector
- PH2.0 Interface (3 foot patch)
- Gain Adjustment Potentiometer
- Power Indicator LED

Precautions

- Before and after use of the pH electrode every time, you need to use (pure)water to clean it.
- The electrode plug should be kept clean and dry in case of short circuit.
- **Preservation**: Electrode reference preservation solution is the **3N KCL** solution.
- Measurement should be avoided staggered pollution between solutions, so as not to affect the accuracy of measurement.
- Electrode blub or sand core is defiled which will make PTS decline, slow response. So, it should be based on the characteristics of the pollutant, adapted to the cleaning solution, the electrode performance recovery.

- Electrode when in use, the ceramic sand core and liquid outlet rubber ring should be removed, in order to make salt bridge solution to maintain a certain velocity.

NOTE: **Differences between the probes, SEN0161 and SEN0169**

Their usages/ specifications are almost the same. The differences locates at

Long-firing Operation: SEN0169 supports, while SEN0161 NOT, i.e. you can not immerse SEN0161 in water for Continuous Testing.

Life Span: In 25 ℃, pure water, do Continuous Testing with them both, SEN0169 can work two years, while SEN0161 can only last for 6 months. And just for reference, if put them in turbid, strongly acid and alkali solution, 25℃, the life span would drop to one year (SEN0169), 1 month(or shorter, SEN0161).
Tempreture, pH, turbidity of the water effect the probe life span a lot.

Waterproof: You can immerse the whole probe SEN0169 into the water, while you can only immerse the front part of the probe SEN0161, the electrode glass bulb, into water, the rear part, from the white shell to the cable, MUST NOT be under water.

Strongly Acid and Alkali: SEN0169 are preferred for strongly acid and alkali test. And if your testing range is usually within pH6~8, then SEN0161 is capable for that.

pH Electrode Characteristics

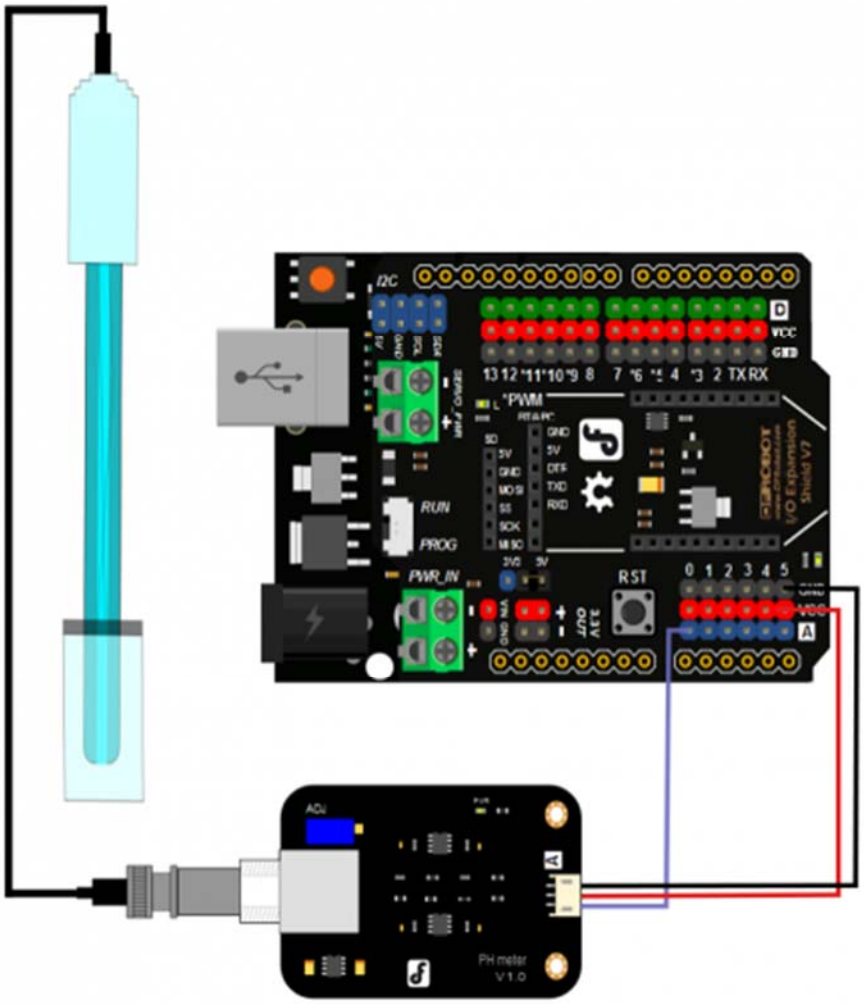
The output of pH electrode is Millivolts, and the pH value of the relationship is shown as follows (25 ℃):

VOLTAGE （mV）	pH value	VOLTAGE （mV）	pH value
414.12	0.00	-414.12	14.00
354.96	1.00	-354.96	13.00
295.80	2.00	-295.80	12.00
236.64	3.00	-236.64	11.00
177.48	4.00	-177.48	10.00
118.32	5.00	-118.32	9.00
59.16	6.00	-59.16	8.00
0.00	7.00	0.00	7.00

NOTE: It is normal that if your reading is much different with the table since you are not reading from the electrode directly but from the voltage adapter, it has converted the original voltage (-5V ~ +5V) to Arduino compatible voltage, i.e. 0 ~ 5V. See the discussion on Forum.

Usage

Connecting Diagram

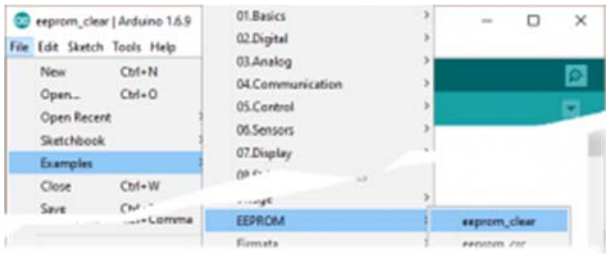


NOTE:

Before you insert the pH probe into one solution from another, or after you finish using the sensor, you must wash the pH electrode with pure water everytime (distilled water is the best)!
The closer **power supply** to +5.00V, the more accurate pH readings you could get.
You have to immerse the pH probe into stationary solution instead of the running one to get relative stable pH readings.
How long should it be under the solution? It depends on the pH value, the closer to neutral solution (pH = 7.00), the longer it will take. As we tested in water pH = 6.0, the blue one costs 6 minutes, and in standard Acid/ Alkali (4.00/ 10.00) solutions, it only needs 10 seconds.

Method 1. Software Calibration

The software calibration is easier than the next part - Hardware Calibration through the Potentiometer. Because it writes the calibration values into Arduino's EEPROM, so you can calibrate once for all if you won't replace your Arduino. It uses mathematical method that to draw a line using two points, i.e. using the Acid standard solution, pH = 4.00 and alkaline pH = 10.00 or 9.18 to draw the linear relation between the voltage and the pH value.



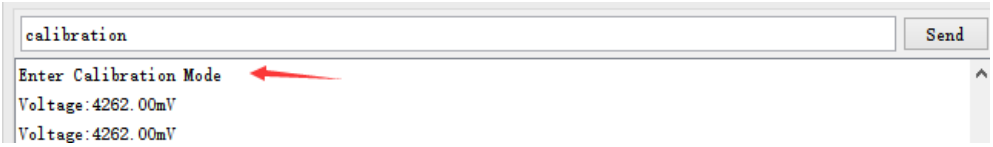
For NOTE 3. Arduino sample sketch "**EEPROM Clear**"

NOTE:

During the calibration (from step 4 to step 7), **power outage** should be avoided, or you will have to start over from step 4.
Software Calibration has nothing to do with the **potentiometer** on the adapter. Especially after you finished the calibration, you should never adjust the potentiometer, or you should start over. Moreover, considering the mechanical vibration might interfere the potentiometer value, you could seal it by Hot Melt Adhesive.
If you want to try Hardware Calibration, you'd better reset the EEPROM setting by uploading the Arduino IDE sample sketch "**EEPROM Clear**" as shown as the right hand picture.

Steps

1. Wiring the pH probe, pH meter adapter (the little PCB board) and Arduino UNO as the Diagram section above.
2. Upload the sample code "Software Calibration" below to UNO.
3. Open Serial Monitor, choose command format as “Both NL & CR”and 115200.
4. Send “**Calibration**” to enter Calibration Mode, and you will see “Enter Calibration Mode” directly.



5. Acid Calibration

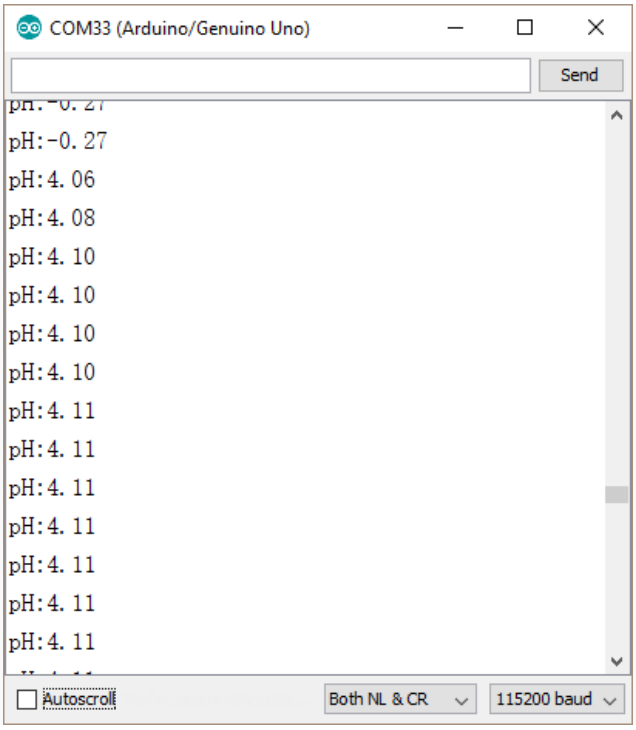
1. Wash your pH probe with pure water (distilled water is best) and dryer it in case of diluting the standard pH solution. Insert it into standard acid solution of pH = 4.0. Wait several seconds till the readings get relative stable.
2. Enter “**acid:4.00**”(no bland space, lower case), and you will get “Acid Calibration Successful” notice. Then go on with Alkali Calibration.

6. Alkali Calibration

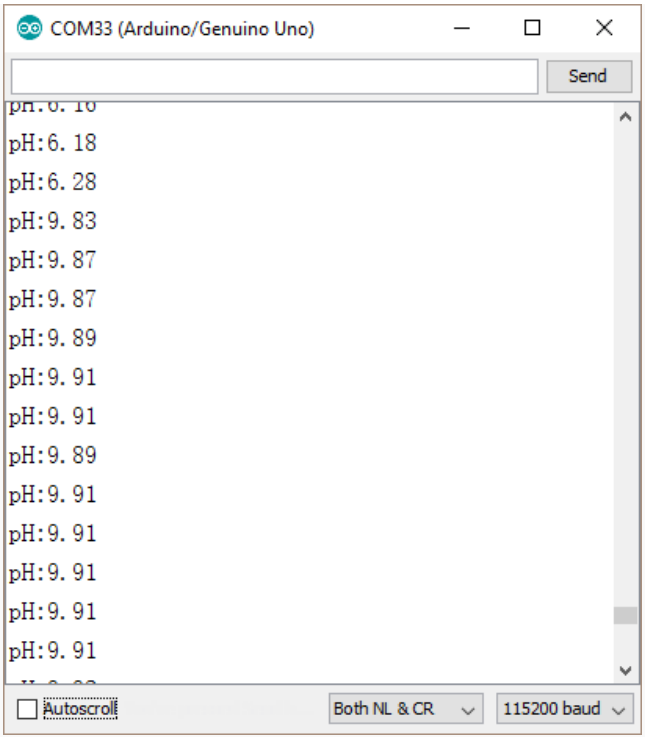
1. Take out the pH probe out of the acid solution, CLEAN it again as you did in last step. After this, insert it into the standard alkali solution with pH = 10 or 9.18. Waiting for the stable readings
2. Enter “**alkali:10.00**”, and you will see “Alkali Calibration Successful”.

7. Enter “exit” to finish calibration. You will see “Calibration Successful,Exit Calibration Mode”.

8. Check if the pH meter was calibrated successfully with the solution pH = 4.00, 9.18, 10.00, if the readings are within the error of 0.1. Congrats!

- 

In Standard acid solution pH = 4.00

- 

In Standard alkali solution pH = 10.00

Sample code: Software Calibration

```
/*************************************************

 This example uses software solution to calibration the ph meter,
not the potentiometer. So it is more easy to use and calibrate.

 This is for SEN0161 and SEN0169.


 Created 2016-8-11

 By youyou from DFrobot <youyou.yu@dfrobot.com>


 GNU Lesser General Public License.

 See <http://www.gnu.org/licenses/> for details.

 All above must be included in any redistribution

 *************************************************/


/***********Notice and Troubleshooting***************

 1.Connection and Diagram can be found here  http://www.dfrobot.co
m/wiki/index.php/PH_meter%28SKU:_SEN0161%29

 2.This code is tested on Arduino Uno.

 *************************************************/


#include <EEPROM.h>

#define EEPROM_write(address, p) {int i = 0; byte *pp = (byte*)&(p
);for(; i < sizeof(p); i++) EEPROM.write(address+i, pp[i]);}

#define EEPROM_read(address, p)  {int i = 0; byte *pp = (byte*)&(p
);for(; i < sizeof(p); i++) pp[i]=EEPROM.read(address+i);}


#define ReceivedBufferLength 20

char receivedBuffer[ReceivedBufferLength+1];   // store the serial
command

byte receivedBufferIndex = 0;
```

```
#define SCOUNT  30           // sum of sample point

int analogBuffer[SCOUNT];    //store the sample voltage

int analogBufferIndex = 0;


#define SlopeValueAddress 0     // (slope of the ph probe)store at the beginning of the EEPROM. The slope is a float number,occupies 4 bytes.

#define InterceptValueAddress (SlopeValueAddress+4)

float slopeValue, interceptValue, averageVoltage;

boolean enterCalibrationFlag = 0;


#define SensorPin A0

#define VREF 5000  //for arduino uno, the ADC reference is the pow er(AVCC), that is 5000mV


void setup()

{

  Serial.begin(115200);

  readCharacteristicValues(); //read the slope and intercept of th e ph probe

}


void loop()

{

  if(serialDataAvailable() > 0)

  {

      byte modeIndex = uartParse();

      phCalibration(modeIndex);    // If the correct calibration c ommand is received, the calibration function should be called.

      EEPROM_read(SlopeValueAddress, slopeValue);     // After cal ibration, the new slope and intercept should be read ,to update cu rrent value.

      EEPROM_read(InterceptValueAddress, interceptValue);

  }
```

```
   static unsigned long sampleTimepoint = millis();
   if(millis()-sampleTimepoint>40U)
   {
     sampleTimepoint = millis();
     analogBuffer[analogBufferIndex] = analogRead(SensorPin)/1024.
0*VREF;    //read the voltage and store into the buffer,every 40ms
     analogBufferIndex++;
     if(analogBufferIndex == SCOUNT)
         analogBufferIndex = 0;
     averageVoltage = getMedianNum(analogBuffer,SCOUNT);   // read
the stable value by the median filtering algorithm
   }


   static unsigned long printTimepoint = millis();
   if(millis()-printTimepoint>1000U)
   {
     printTimepoint = millis();
     if(enterCalibrationFlag)             // in calibration mode,
print the voltage to user, to watch the stability of voltage
     {
       Serial.print("Voltage:");
       Serial.print(averageVoltage);
       Serial.println("mV");
     }else{
     Serial.print("pH:");              // in normal mode, print th
e ph value to user
     Serial.println(averageVoltage/1000.0*slopeValue+interceptValu
e);
     }
   }
}


boolean serialDataAvailable(void)
{
```

```
  char receivedChar;
  static unsigned long receivedTimeOut = millis();
  while (Serial.available()>0)
  {
    if (millis() - receivedTimeOut > 1000U)
    {
      receivedBufferIndex = 0;
      memset(receivedBuffer,0,(ReceivedBufferLength+1));
    }
    receivedTimeOut = millis();
    receivedChar = Serial.read();
    if (receivedChar == '\n' || receivedBufferIndex==ReceivedBuffe
rLength){
                receivedBufferIndex = 0;
                strupr(receivedBuffer);
                return true;
    }
    else{
      receivedBuffer[receivedBufferIndex] = receivedChar;
      receivedBufferIndex++;
    }
  }
  return false;
}


byte uartParse()
{
  byte modeIndex = 0;
  if(strstr(receivedBuffer, "CALIBRATION") != NULL)
      modeIndex = 1;
  else if(strstr(receivedBuffer, "EXIT") != NULL)
      modeIndex = 4;
  else if(strstr(receivedBuffer, "ACID:") != NULL)
```

```
      modeIndex = 2;
  else if(strstr(receivedBuffer, "ALKALI:") != NULL)
      modeIndex = 3;
  return modeIndex;
}


void phCalibration(byte mode)
{
    char *receivedBufferPtr;
    static byte acidCalibrationFinish = 0, alkaliCalibrationFinish
= 0;
    static float acidValue,alkaliValue;
    static float acidVoltage,alkaliVoltage;
    float acidValueTemp,alkaliValueTemp,newSlopeValue,newIntercept
Value;
    switch(mode)
    {
      case 0:
      if(enterCalibrationFlag)
         Serial.println(F("Command Error"));
      break;

      case 1:
      receivedBufferPtr=strstr(receivedBuffer, "CALIBRATION");
      enterCalibrationFlag = 1;
      acidCalibrationFinish = 0;
      alkaliCalibrationFinish = 0;
      Serial.println(F("Enter Calibration Mode"));
      break;

      case 2:
      if(enterCalibrationFlag)
      {
```

```
          receivedBufferPtr=strstr(receivedBuffer, "ACID:");
          receivedBufferPtr+=strlen("ACID:");
          acidValueTemp = strtod(receivedBufferPtr,NULL);
          if((acidValueTemp>3)&&(acidValueTemp<5))        //typica
l ph value of acid standand buffer solution should be 4.00
          {
             acidValue = acidValueTemp;
             acidVoltage = averageVoltage/1000.0;        // mV ->
V
             acidCalibrationFinish = 1;
             Serial.println(F("Acid Calibration Successful"));
           }else {
             acidCalibrationFinish = 0;
             Serial.println(F("Acid Value Error"));
           }
      }
      break;

       case 3:
       if(enterCalibrationFlag)
       {
           receivedBufferPtr=strstr(receivedBuffer, "ALKALI:");
           receivedBufferPtr+=strlen("ALKALI:");
           alkaliValueTemp = strtod(receivedBufferPtr,NULL);
           if((alkaliValueTemp>8)&&(alkaliValueTemp<11))        //
typical ph value of alkali standand buffer solution should be 9.18
or 10.01
           {
                 alkaliValue = alkaliValueTemp;
                 alkaliVoltage = averageVoltage/1000.0;
                 alkaliCalibrationFinish = 1;
                 Serial.println(F("Alkali Calibration Successful")
);
            }else{
```

```
                alkaliCalibrationFinish = 0;
                Serial.println(F("Alkali Value Error"));
            }
        }
        break;

         case 4:
         if(enterCalibrationFlag)
         {
             if(acidCalibrationFinish && alkaliCalibrationFinish)
             {
               newSlopeValue = (acidValue-alkaliValue)/(acidVoltage
- alkaliVoltage);
               EEPROM_write(SlopeValueAddress, newSlopeValue);
               newInterceptValue = acidValue - (slopeValue*acidVolt
age);
               EEPROM_write(InterceptValueAddress, newInterceptValu
e);
               Serial.print(F("Calibration Successful"));
             }
             else Serial.print(F("Calibration Failed"));
             Serial.println(F(",Exit Calibration Mode"));
             acidCalibrationFinish = 0;
             alkaliCalibrationFinish = 0;
             enterCalibrationFlag = 0;
         }
         break;
    }
}

int getMedianNum(int bArray[], int iFilterLen)
{
      int bTab[iFilterLen];
      for (byte i = 0; i<iFilterLen; i++)
```

```
      {
          bTab[i] = bArray[i];
      }
      int i, j, bTemp;
      for (j = 0; j < iFilterLen - 1; j++)
      {
          for (i = 0; i < iFilterLen - j - 1; i++)
          {
            if (bTab[i] > bTab[i + 1])
            {
                bTemp = bTab[i];
                bTab[i] = bTab[i + 1];
                bTab[i + 1] = bTemp;
            }
          }
      }
      if ((iFilterLen & 1) > 0)
        bTemp = bTab[(iFilterLen - 1) / 2];
      else
        bTemp = (bTab[iFilterLen / 2] + bTab[iFilterLen / 2 - 1]) /
2;
      return bTemp;
}


void readCharacteristicValues()
{
    EEPROM_read(SlopeValueAddress, slopeValue);
    EEPROM_read(InterceptValueAddress, interceptValue);
    if(EEPROM.read(SlopeValueAddress)==0xFF && EEPROM.read(SlopeVa
lueAddress+1)==0xFF && EEPROM.read(SlopeValueAddress+2)==0xFF && E
EPROM.read(SlopeValueAddress+3)==0xFF)
    {
      slopeValue = 3.5;   // If the EEPROM is new, the recommendat
ory slope is 3.5.
```

```
      EEPROM_write(SlopeValueAddress, slopeValue);
    }
    if(EEPROM.read(InterceptValueAddress)==0xFF && EEPROM.read(Int
erceptValueAddress+1)==0xFF && EEPROM.read(InterceptValueAddress+2
)==0xFF && EEPROM.read(InterceptValueAddress+3)==0xFF)
    {
      interceptValue = 0;  // If the EEPROM is new, the recommenda
tory intercept is 0.
      EEPROM_write(InterceptValueAddress, interceptValue);
    }
}
```

Method 2. Hardware Calibration through potentiometer

If you've taken the Method 1. Software Calibration, you can ignore this part.

1. Connect according to the graphic, that is, the pH electrode is connected to the BNC connector on the pH meter board, and then use the connection lines, the pH meter board is connected to the analog port 0 of the Arduino controller. When the Arduino controller gets power, you will see the blue LED on board is on.
2. Upload the sample code to the Arduino controller.
3. Put the pH electrode into the standard solution whose pH value is 7.00, or directly short circuit the input of the BNC connector. Open the serial monitor of the Arduino IDE, you can see the pH value printed to it, and the error does not exceed 0.3. Record the pH value printed, then compared with 7.00, and the difference should be changed into the "Offset" in the sample code. For example, the pH value printed is 6.88, so the difference is 0.12. You should change the # **define Offset 0.00** into # **define Offset 0.12** in the sample code.
4. **Fine adjustment**

- **For Acid solution:** Put the pH electrode into the pH standard solution whose value is 4.00. Then wait about a minute, adjust the Gain Potential device, let the value stabilise at around 4.00. At this time, the acidic calibration has been completed and you can measure the pH value of an acidic solution.
- **For Alkaline solution:** According to the linear characteristics of pH electrode itself, after the above calibration, you can

directly measure the pH value of the **alkaline** solution, but if you want to get a better accuracy, you can recalibrate it with the standard solution, pH = 9.18. Also adjust the gain potential device, let the value stabilise at around 9.18. After this calibration, you can measure the pH value of the alkaline solution.

Sample Code for Hardware Calibration

```
/*
 # This sample code is used to test the pH meter V1.0.
 # Editor : YouYou
 # Ver    : 1.0
 # Product: analog pH meter
 # SKU    : SEN0161
*/
#define SensorPin A0            //pH meter Analog output to Arduin
o Analog Input 0
#define Offset 0.00            //deviation compensate
#define LED 13
#define samplingInterval 20
#define printInterval 800
#define ArrayLenth  40    //times of collection
int pHArray[ArrayLenth];   //Store the average value of the sensor
feedback
int pHArrayIndex=0;
void setup(void)
{
  pinMode(LED,OUTPUT);
  Serial.begin(9600);
  Serial.println("pH meter experiment!");    //Test the serial mon
itor
}
```

```
void loop(void)
{
  static unsigned long samplingTime = millis();
  static unsigned long printTime = millis();
  static float pHValue,voltage;
  if(millis()-samplingTime > samplingInterval)
  {
      pHArray[pHArrayIndex++]=analogRead(SensorPin);
      if(pHArrayIndex==ArrayLenth)pHArrayIndex=0;
      voltage = avergearray(pHArray, ArrayLenth)*5.0/1024;
      pHValue = 3.5*voltage+Offset;
      samplingTime=millis();
  }
  if(millis() - printTime > printInterval)   //Every 800 milliseco
nds, print a numerical, convert the state of the LED indicator
  {
        Serial.print("Voltage:");
        Serial.print(voltage,2);
        Serial.print("    pH value: ");
        Serial.println(pHValue,2);
        digitalWrite(LED,digitalRead(LED)^1);
        printTime=millis();
  }
}
double avergearray(int* arr, int number){
  int i;
  int max,min;
  double avg;
  long amount=0;
  if(number<=0){
    Serial.println("Error number for the array to avraging!/n");
    return 0;
  }
```

```
  if(number<5){   //less than 5, calculated directly statistics
    for(i=0;i<number;i++){
      amount+=arr[i];
    }
    avg = amount/number;
    return avg;
  }else{
    if(arr[0]<arr[1]){
      min = arr[0];max=arr[1];
    }
    else{
      min=arr[1];max=arr[0];
    }
    for(i=2;i<number;i++){
      if(arr[i]<min){
        amount+=min;        //arr<min
        min=arr[i];
      }else {
        if(arr[i]>max){
          amount+=max;    //arr>max
          max=arr[i];
        }else{
          amount+=arr[i]; //min<=arr<=max
        }
      }//if
    }//for
    avg = (double)amount/(number-2);
  }//if
  return avg;
}
```

FAQ

Q1. My PH sensor readings are not correct, what did I miss?

Or the module is defective?

A. 1. Check if the pH sensor circuit board is good? Read on the Forum. or on wiki for the steps. During the transport, there might be crash causing the probe head cracked, please check if the probe is good or not.

2. If you don't use Arduino as the controller, then please check your ADC module that whether it converts the 5V analog input to 1024, if it is 4096(or other byte), please re-determine the equation in the code.

Q2. Big fluctuations in ph meter readings. When I make measurements in a glass, I have correct, stable reading. But when I put it inside the aquarium with the pumping system working, the easurement varies even more than a degree, and it's not stable, if I swicth off the pump the given value doesn´t oscilate anymore.

A. There should be NO working electrical device in the container. Any tiny leakage of electricity will cause the probe working error. Especially, many people bought the EC meter and put it into the same tank for the test, but then the pH meter cannot work well anymore. Please seperate them into different containers, or turning off the EC meter when using the pH meter.

Q3. May I know the Maximum range different if we do not calibrate the pH meter.

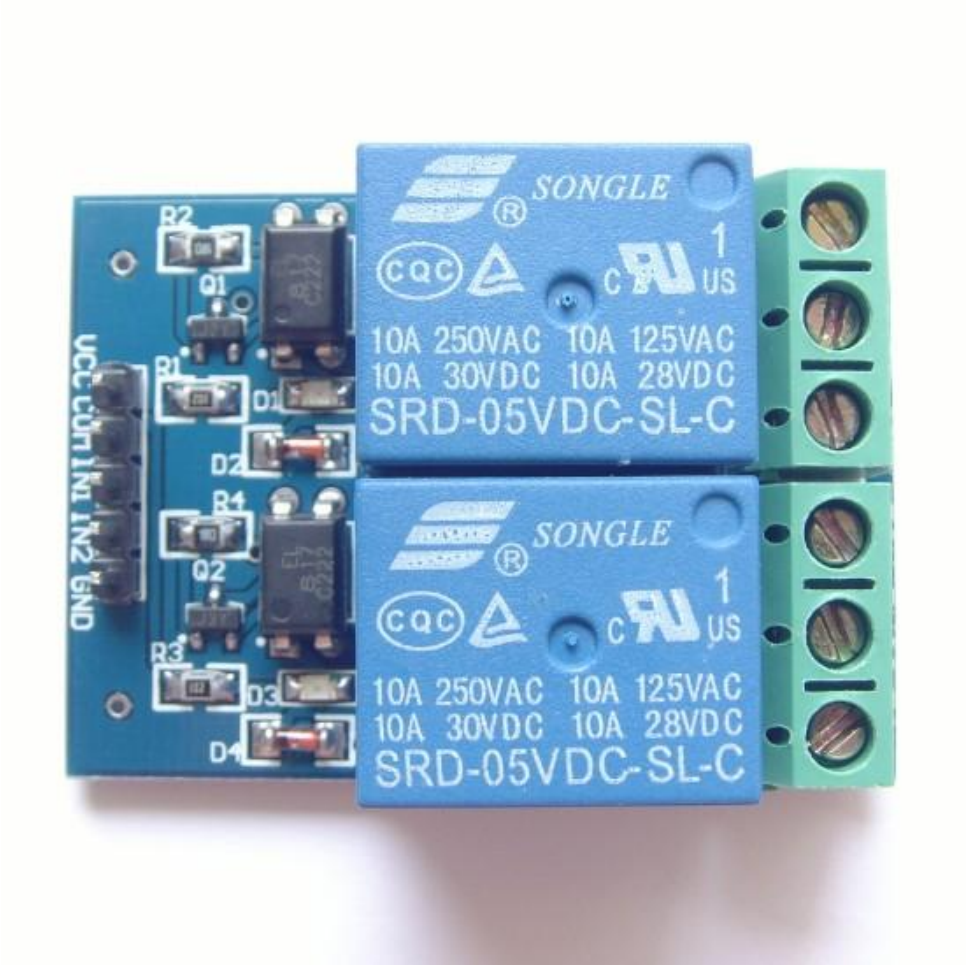
A. The maximum range differs from probe, you have to calibrate it before use if the pH probe was kept long.

Q4. I would just like to ask if your pH sensor can be connect to any micro controller aside from arduino. Would it be compatible with a raspberry pi? Thank You!

A. Yes, it can be used on any device as long as it could give 5V power supply and accept 5V analog signal, but as the Rasp pi is only compatible with 3.3V sensor, so an expansion shield is suggested to use with (please make sure which kind of Pi you use)

For any questions and more cool ideas to share, please visit **DFRobot Forum**

2 CHANNEL 5V 10A RELAY MODULE



Description

The relay module is an electrically operated switch that allows you to turn on or off a circuit using voltage and/or current much higher than a microcontroller could handle. There is no connection between the low voltage circuit operated by the microcontroller and the high power circuit. The relay protects each circuit from each other.

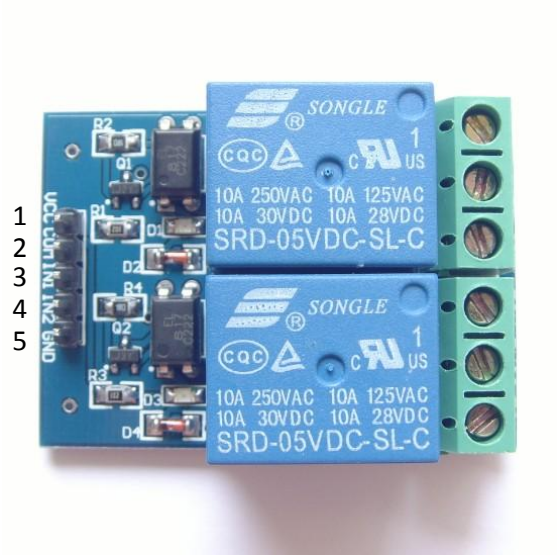
The each channel in the module has three connections named NC, COM, and NO. Depending on the input signal trigger mode, the jumper cap can be placed at high

level effective mode which ‘closes’ the normally open (NO) switch at high level input and at low level effective mode which operates the same but at low level input.

Specifications

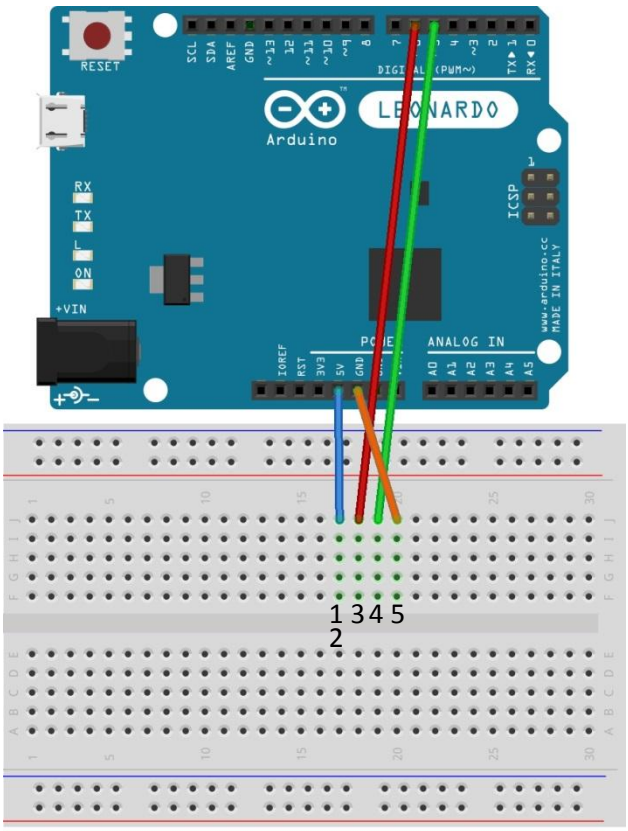
- On-board EL817 photoelectric coupler with photoelectric isolating anti-interference ability strong
- On-board 5V, 10A / 250VAC, 10A / 30VDC relays
- Relay long life can absorb 100000 times in a row
- Module can be directly and MCU I/O link, with the output signal indicator
- Module with diode current protection, short response time
- PCB Size: 45.8mm x 32.4mm

Pin Configuration

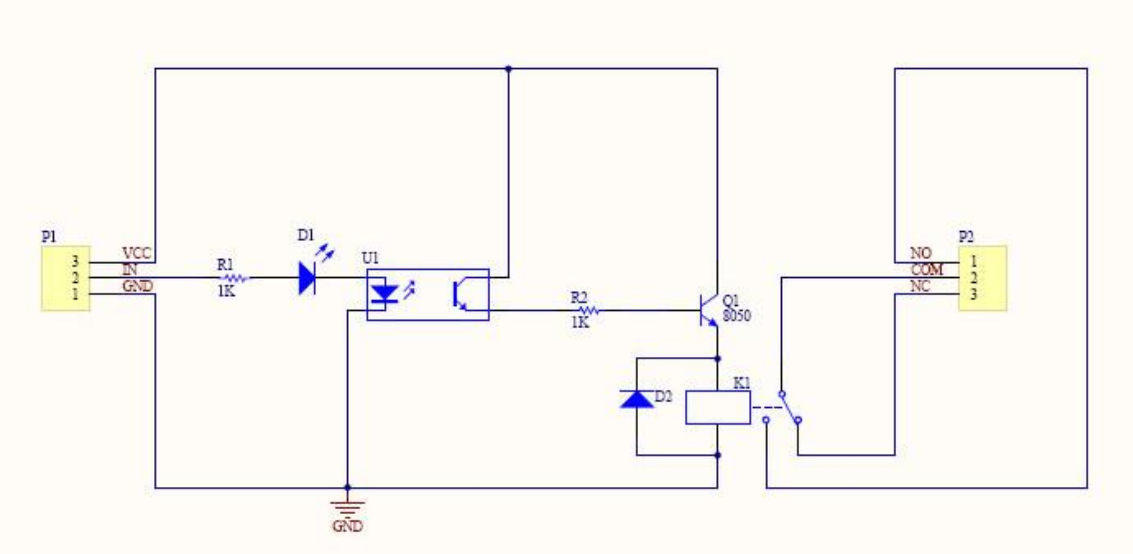


1. VCC: 5V DC
2. COM: 5V DC
3. IN1: high/low output
4. IN2: high/low output
5. GND: ground

Wiring Diagram



Schematic Diagram



Sample Sketch

```
void setup(){
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
}

void loop(){
  digitalWrite(5, LOW);
  digitalWrite(6, HIGH);
  delay(4000);
  digitalWrite(5, HIGH);
  digitalWrite(6, LOW);
  delay(4000);
}
```

How to Test

The components to be used are:

- Microcontroller (any compatible arduino)
- 2 channel 5V 10A relay module
- Pin connectors
- Breadboard
- USB cable

1. Connect the components based on the figure shown in the wiring diagram using pin connectors. VCC and COM pin is connected to the 5V power supply, GND pin is connected to the GND, IN1 and IN2 pins are connected to the digital I/O pin. Pin number will be based on the actual program code.
2. After hardware connection, insert the sample sketch into the Arduino IDE.
3. Using a USB cable, connect the ports from the microcontroller to the computer.
4. Upload the program.

Testing Results

The figures below shows an alternate switching of the two relays every 4 seconds. A tick sound and a red LED would be observed.