● *Abstract:*

In this Arduino based automatic water level controller project we are going to measure the water level by using ultrasonic sensors. Basic principle of ultrasonic distance measurement is based on ECHO. When sound waves are transmitted in environment then they return to the origin as ECHO after striking on any obstacle. So, we must only calculate its traveling time of both sounds means outgoing time and returning time to origin after striking on any obstacle. And after some calculation we can get a result that is the distance. This concept is used in our water controller project where the water motor pump is automatically turned on when water level in the tank becomes low. You can also check this simple water level indicator circuit for a simpler version of this project.

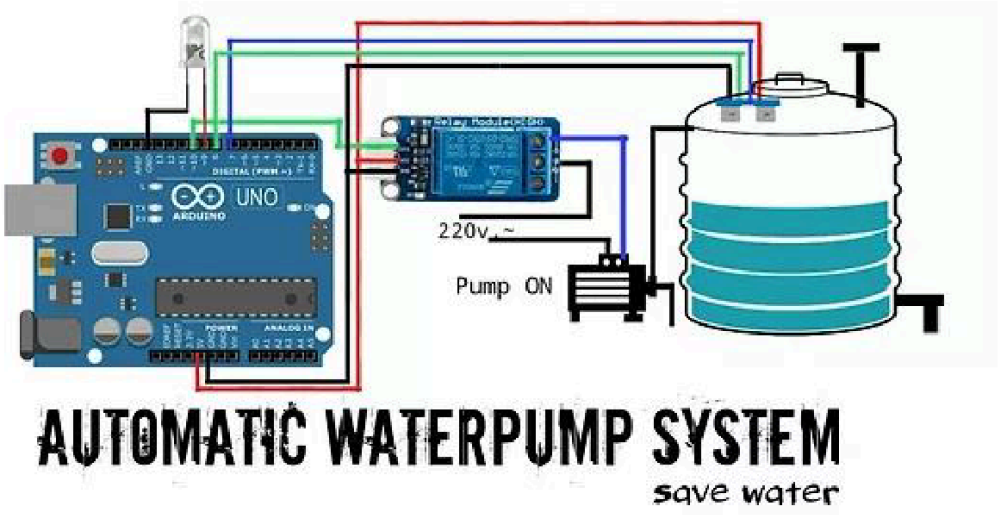
● *Introduction:*

The automatic water control pump circuit posted here is used to automate the operation of an electrical water pump based on the level of water in the overhead tank. The automatic water pump controller using Arduino circuit can be used as a standalone system and can be interfaced to the existing control panel. In this project we have used very common, efficient, and cheap components. It is very simple circuit and useful for all. It is used in home, public forums, industries. Especially, homes with elderly people it is a necessary one to fill the tank for daily purpose. And for daily office workers it is one needful to fill the tank.

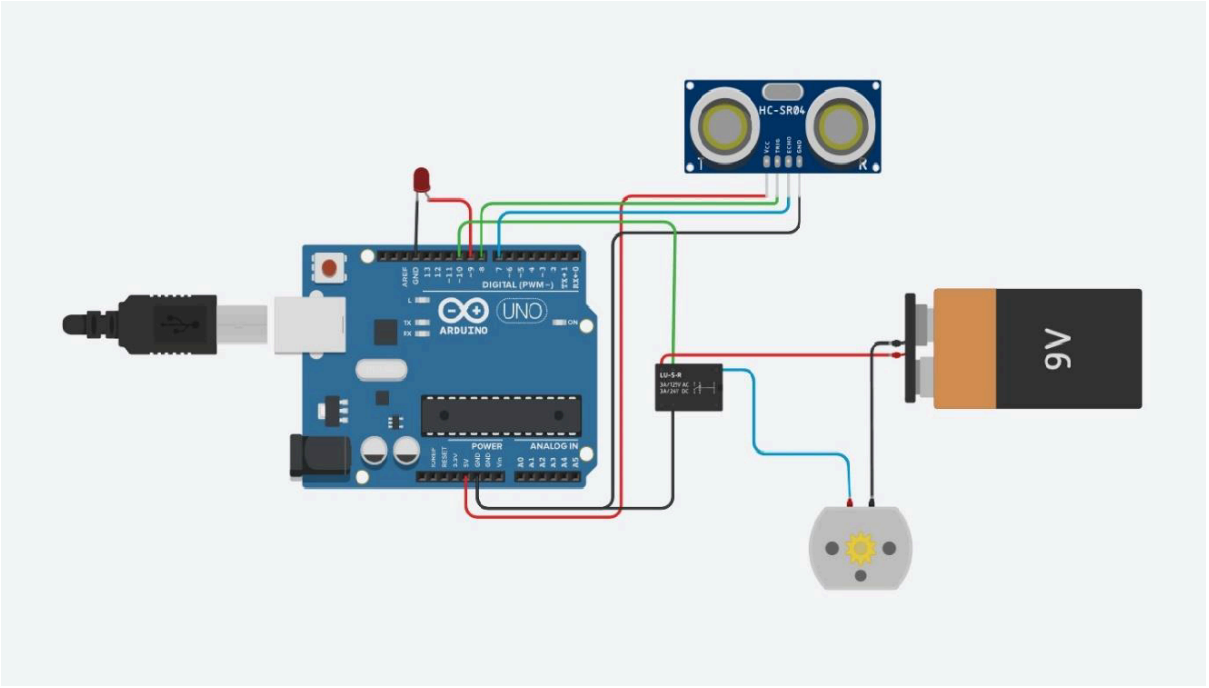
● *Objective*

- The main motive of this project is to save water and electricity.
- This project does not require any human to operate as it is fully automatic.
- This project can be connected in real life starter and control a motor.
- The operating current of this circuit is very small and it does not waste any power and does not produce any heat.
- Though it is automatic, it can be controlled using manual switches as per our convenient.

● *System Architechture:*

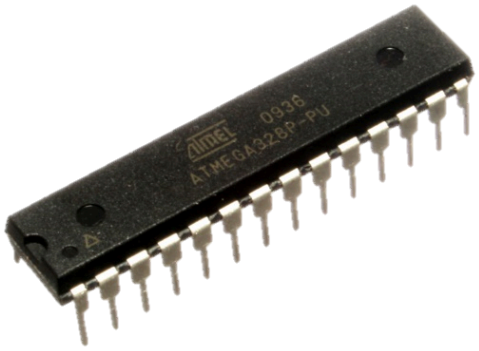


● *Circuit Design:*



● *Description of the Instruments used:*

1. ATmega328:



ATMega328 is the ATMEL Microcontroller on which Arduino UNO is based. The Atmel 8-bit AVR RISC-based microcontroller combines 32 kB ISP flash memory with read-while-write capabilities, 1 kB EEPROM, 2 kB SRAM, 23 general purpose I/O lines, 32 general purpose working registers, three flexible timer/counters with compare modes, internal and external interrupts, serial programmable USART, a byte-oriented 2-wire serial interface, SPI serial port, 6-channel 10-bit A/D converter (8-channels in TQFP and QFN/MLF packages), programmable watchdog timer with internal oscillator, and five software selectable power saving modes. The device operates between 1.8-5.5 volts The device achieves throughput approaching 1 MIPS per MHz Serial data to the MCU is clocked on the rising edge and data from the MCU is clocked on the falling edge. Power is applied to VCC while RESET and SCK are set to zero.

2. *Ultrasonic Sensor:*

Ultrasonic Sensor Pinout Configuration:

Pin Number	Pin Name	Description
1	V_{cc}	The V_{cc} pin powers the sensor, typically with +5V
2	Trigger	Trigger pin is an Input pin. This pin must be kept high for 10us to initialize measurement by sending US wave.
3	Echo	Echo pin is an Output pin. This pin goes high for a period of time which will be equal to the time taken for the US wave to return back to the sensor.
4	Ground	This pin is connected to the Ground of the system.

HC-SR04 Sensor Features:

- Operating voltage: +5V
- Theoretical Measuring Distance: 2cm to 450cm
- Practical Measuring Distance: 2cm to 80cm
- Accuracy: 3mm
- Measuring angle covered: <15°
- Operating Current: <15mA
- Operating Frequency: 40Hz

HC-SR04 Ultrasonic Sensor – Working:

As shown above the HC-SR04 Ultrasonic (US) sensor is a 4-pin module, whose pin names are V_{cc}, Trigger, Echo, and Ground respectively. This sensor is a very popular sensor used in many applications where measuring distance or sensing objects are required. The module has two eyes like projects in the front which forms the Ultrasonic transmitter and Receiver. The sensor works with the simple high school formula that

Distance = Speed × Time

The Ultrasonic transmitter transmits an ultrasonic wave, this wave travels in air and when it gets objected by any material it gets reflected back toward the sensor this reflected wave is observed by the Ultrasonic receiver module as shown in the picture below.

Now, to calculate the distance using the above formulae, we should know the Speed and time. Since we are using the Ultrasonic wave we know the universal speed of US wave at room conditions which is 330m/s. The circuitry inbuilt on the module will calculate the time taken for the US wave to come back and turns on the echo pin high for that same particular amount of time, this way we can also know the time taken. Now simply calculate the distance using a microcontroller or microprocessor.

How to use the HC-SR04 Ultrasonic Sensor?

HC-SR04 distance sensor is commonly used with both microcontroller and microprocessor platforms like Arduino, ARM, PIC, Raspberry Pie etc. The following guide is universally since it has to be followed irrespective of the type of computational device used.

Power the Sensor using a regulated +5V through the Vcc ad Ground pins of the sensor. The current consumed by the sensor is less than 15mA and hence can be directly powered by the on board 5V pins (If available). The Trigger and the Echo pins are both I/O pins and hence they can be connected to I/O pins of the microcontroller. To start the measurement, the trigger pin has to be made high for 10uS and then turned off. This action will trigger an ultrasonic wave at frequency of 40Hz from the transmitter and the receiver will wait for the wave to return. Once the wave is returned after it getting reflected by any object the Echo pin goes high for a particular amount of time which will be equal to the time taken for the wave to return back to the sensor.

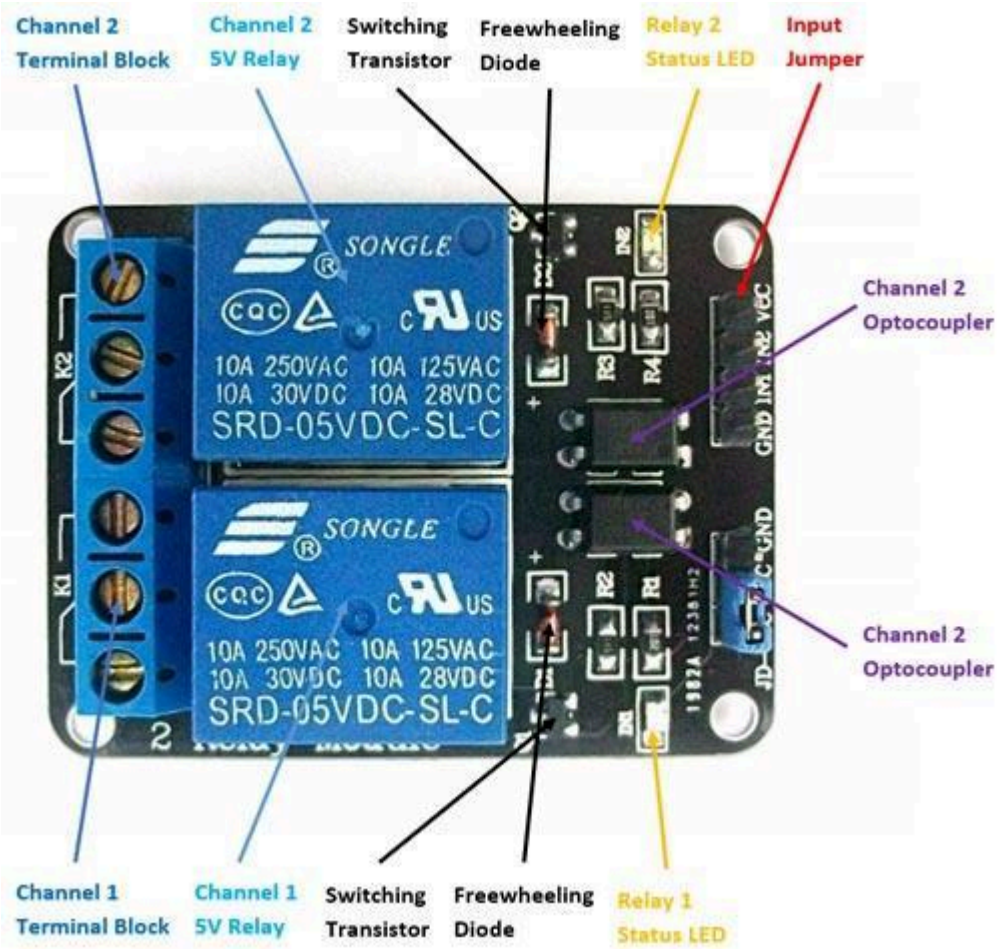
The amount of time during which the Echo pin stays high is measured by the MCU/MPU as it gives the information about the time taken for the wave to return back to the Sensor. Using this information, the distance is measured as explained in the above heading.

Applications:

- Used to avoid and detect obstacles with robots like biped robot, obstacle avoider robot, path finding robot etc.
- Used to measure the distance within a wide range of 2cm to 400cm

- Can be used to map the objects surrounding the sensor by rotating it
- Depth of certain places like wells, pits etc can be measured since the waves can penetrate through water.

3. 2-Channel 5V Relay Module:



2-Channel 5V Relay Module is a relay interface board, it can be controlled directly by a wide range of microcontrollers such as Arduino, AVR, PIC, ARM and so on. It uses a low-level triggered control signal (3.3-5VDC) to control the relay. Triggering the relay operates the normally open or normally closed contacts. It is frequently used in an automatic control circuit. To put it simply, it is an automatic switch to control a high-current circuit with a low-current signal.5V relay signal input voltage range, 0-5V. VCC power to the system. JD-VCC relay in the power supply. JD-VCC and VCC can be a shorted.

The features of 2-Channel Relay module:

- Good for safe control of higher amperage circuits. In power systems, the lower current can control the higher one.
- 2-channel high voltage system output, meeting the needs of dual channel control.
- Brand new and high quality.
- Standard interface that can be controlled directly by microcontroller (Arduino, 8051, AVR, PIC, DSP, ARM)]
- Wide range of controllable voltages.
- Being able to control high load current, which can reach 250V, 10A or 125V, 15A
- With a normally-open (NO) contact and a normally-closed (NC) contact.
- Around the board with 4 mounting holes, easy installation and fixing
- It has a common end, a beginning, a closed-end

Specification of 2-Channel Relay module:

- Relay Module: Model- JQC-3FF-S-Z, 2 Channel
- Colour: Blue Relays on a black PCB
- Load: 10A, AC 250V/ 15A, 125V
- Supply voltage: 3.75V to 6V
- Trigger current: 5mA
- Current when relay is active: ~70mA (single), ~140mA (both)
- Relay maximum contact voltage: 250VAC, 30VDC
- Relay maximum current: 10A

Dual-Channel Relay Module Pinout:

Pin Number	Pin Name	Description
1	JD-V_{CC}	Input for isolated power supply for relay coils

2	V_{cc}	Input for directly powering the relay coils
3	GND	Input ground reference
4	GND	Input ground reference
5	IN1	Input to activate the first relay
6	IN2	Input to activate the second relay
7	V_{cc}	V_{cc} to power the optocouplers, coil drivers, and associated circuitry

Components present on a 5V Dual Channel Relay Module:

The following are the major components that would find on a Dual channel Relay module, we will discuss them in detail further in this article. 5V Relay, Optocoupler, Diodes, Transistors, Resistors, LEDs, Male Headers, 3-pin Terminal connectors, etc.

The dual-channel relay module contains switching relays and the associated drive circuitry to make it easy to integrate relays into a project powered by a microcontroller. On the left are two terminal blocks, which are used to connect mains wires to the module without soldering.

Next, come to the two relays. As marked on the body of the relay, the relay coil is rated for 5VDC, and the contacts are rated for 10A at 250VAC or 30VDC, or 125VAC or 28VDC.

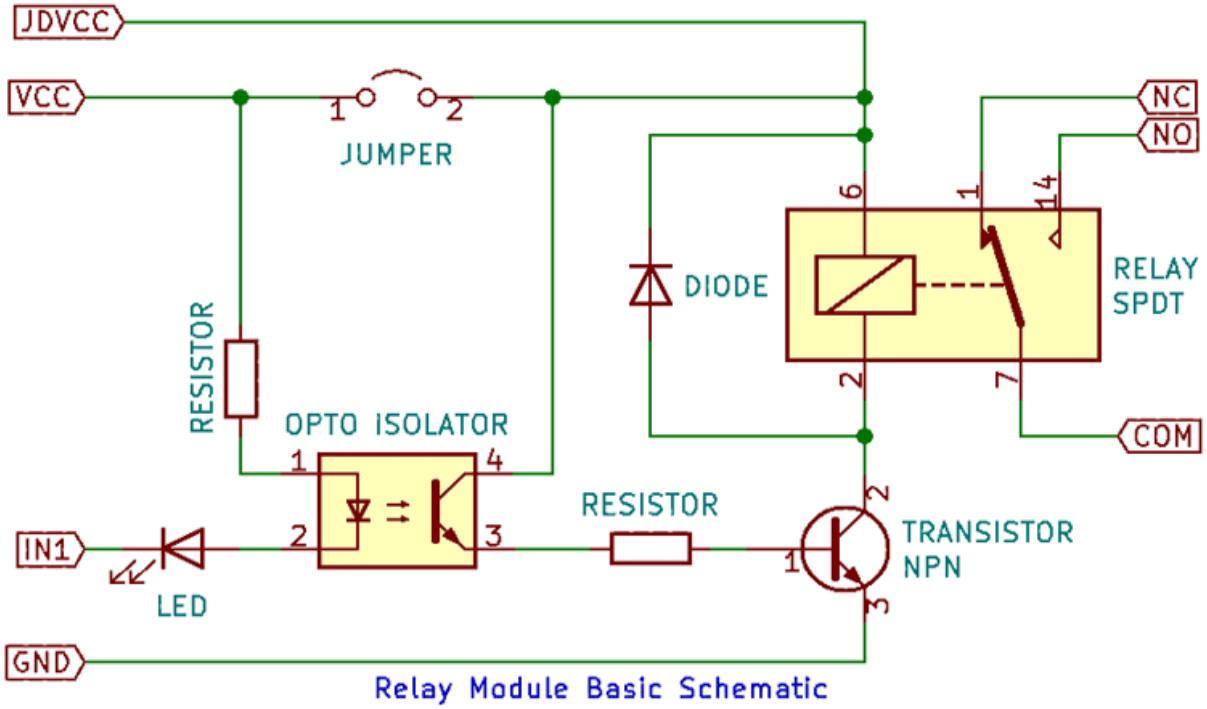
The switching transistors amplify the signal from the inputs enough to drive the relay. The freewheeling diodes prevent voltage spikes across the switching transistors. The status LEDs turn on when the relay is active and indicate switching.

The optocouplers are used to provide additional isolation between the input and the relays. The isolation can be selected using the VCC/JDVCC jumper.

The input jumper has two input and two power pins and can be easily used to connect to jumper wires and other microcontrollers and sensors.

Internal Circuit Diagram for Dual-Channel Relay Module:

This same circuit is replicated twice on the module. There are several differences when compared to the single-channel relay module. The first is that the optocoupler and relay can have a separate power supply from the input. This is selected using the JD-VCC jumper – if the jumper is shorted, then the relay coil and optocoupler output are connected to the signal VCC. If the jumper is left open, a separate power supply can be provided to the relay coil through the JD-VCC pin.



The second is that the input is active low, meaning that the relay coil is activated only when the input is low. This is because the optocoupler input and the indicator LED are chained to V_{CC}. When the input is high, the voltage difference across the chain is 0V and no current can flow, but when the input is low, supply voltage appears across the chain and the optocoupler output conducts, supplying current to the base of the drive transistor, turning the relay on.

Dual-Channel Relay Module Basic Troubleshooting:

If either of the relays does not turn on:

- The contacts might be welded due to overcurrent/arcing. Shaking the module firmly might help unstick the contacts
- The driver circuitry might have been damaged due to overvoltage.
- Input polarity might be incorrect.
- Jumper might not have been moved to the correct position

Dual-Channel Relay Module Applications:

- Switching mains loads
- Home automation
- Battery backup
- High current load switching

4. _5V DC Water Pump/Submersible Pump:

Description:

5V DC Water Pump for Arduino is a low cost, small size Submersible Water Pump which can be operated from a 2.5 ~ 6V power supply.

It can take up to 120 litres per hour with a very low current consumption of 220mA.

The main advantages of this water pump are their operating current. Because it is so low, they can be powered directly by the Arduino 5V outputs, making a very simple set-up.

This will make it easier and faster for prototyping of your automated watering system.

Just connect tube pipe to the motor outlet, submerge it in water and power it. Make sure that the water level is always higher than the DC Pump. Dry run may damage the motor due to heating and it will also produce noise.

This 5v dc water pump is cheap and useful for prototyping but they won't last very long under intensive work load. So, they are good for prototyping and projects that require watering from time to time and not a continuous water flow.

Finding a reliable 5v submersible water pump that can work at the same voltage than Arduino not only is possible it is probably the best option for building automatic plant watering projects.

Water Pump for Arduino is useful for controlling water flow. Paired with an Arduino microcontroller, these pumps can automate the flow distribution process, which can be useful for applications such as aquariums, fountains, and systems for irrigation, refilling and aeration.

Mini water pump 5V is a Miniature water Pump. This is a 5V DC motor driven water pump. This Submersible water Pump Can use with All Micro controller units including Arduino, Raspberry Pi, Node MCU.

Features:

- DC 3v to 6v submersible pump

- micro mini submersible water pump 3v to 6v
- DC water pump for DIY
- DC pump for HOBBY kit.
- Stable and reliable performance.
- Easy disassembly type.
- Easy to clean and maintain.
- Long service life.

5V DC Water Pump for Arduino Specifications:

- Operating Current: 130 ~ 220mA
- Operating Voltage: 2.5 ~ 6V
- Flow Rate: 80 ~ 120 L/H
- Maximum Lift: 40 ~ 110 mm
- Driving Mode: DC, Magnetic Driving
- Continuous Working Life: 500 hours
- Material: Engineering Plastic
- Outlet Outside Diameter: 7.5 mm
- Outlet Inside Diameter: 5 mm

5. 9V Battery and Connector:

The 9V batteries are extremely well-known batteries that was first used in transistor radios. Possible chemistries of primary (non-rechargeable) 9V batteries include Alkaline, carbon-Zinc, Lithium. Possible chemistries of secondary (rechargeable) 9 V batteries include Nickel-Cadmium, Nickel-Metal hydride (NiMH) and Lithium ion.

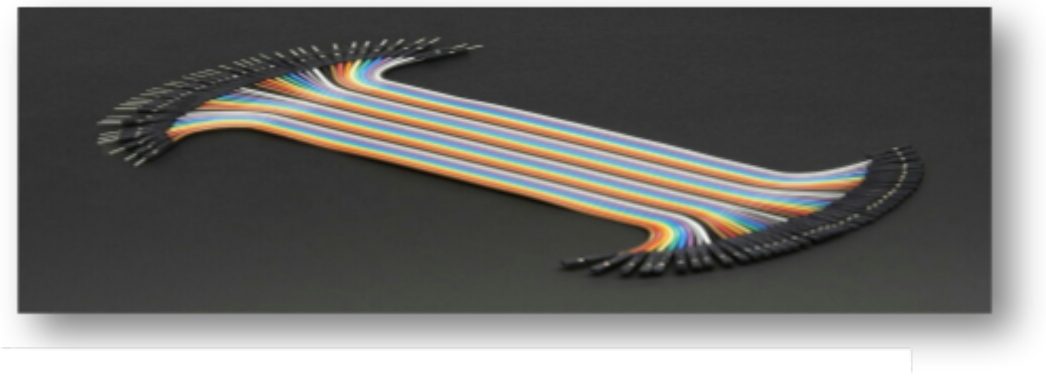
The PP3 batteries have both terminals in a snap connector on one end. The smaller circular (male) terminal is positive, and the larger hexagonal or octagonal (female) terminal is the negative contact.

9V Battery Specifications

Specifications	Series	Manufacturer

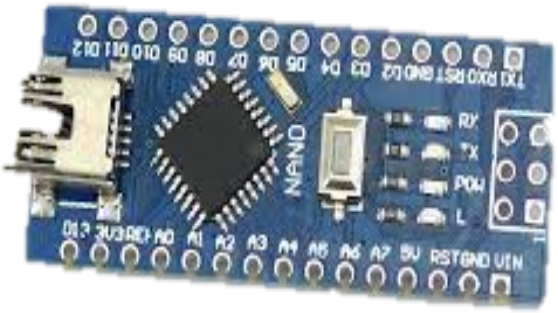
Normal Voltage: 9V Capacity (Alkaline) ~ 550 mAh Capacity (NiMH) ~ 175-300 mAh Capacity (Cabon-Zinc) ~ 400 mAh Operating Temperature: 0° C - 60° C Dimension: 17.5 mm X 48.5 mm X 26.5 mm	6F22	HYW

6. Jumper Wires:



A jump wire, a.k.a jumper, jumper wire, DuPont wire is an electrical wire, or group of them in a cable, with a connector or pin at each end, which is normally used to interconnect the components of a breadboard or other prototype or test circuit, internally or with other equipment or components, without soldering. Jumpers typically come in three versions: male-to-male, male-to-female, female-to-female based on the differences between each is in end point of the wire.

7. Arduino Nano:



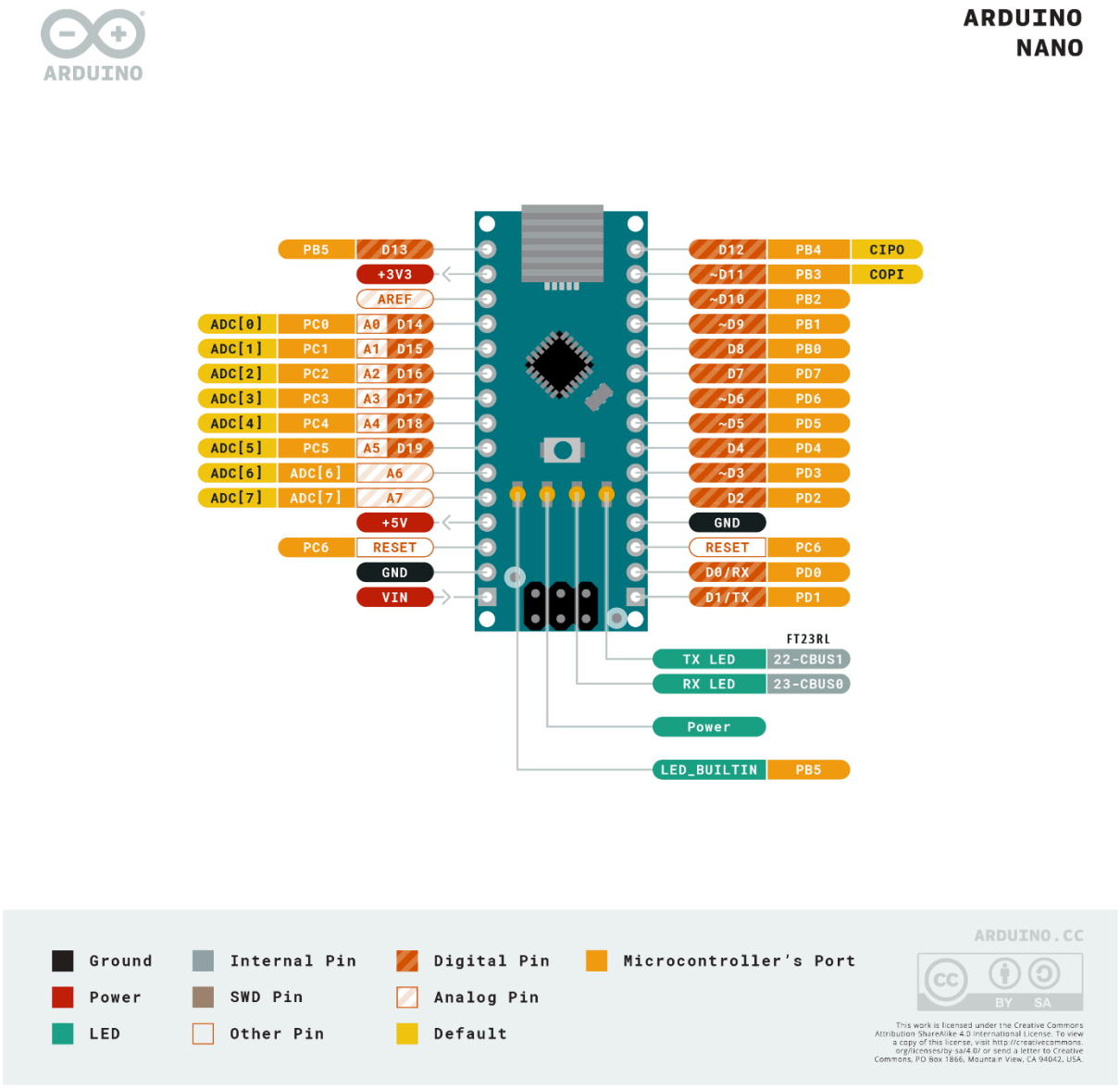
The Arduino Nano is a small, complete, and breadboard-friendly board based on the ATmega328 (Arduino Nano 3.x). It has more or less the same functionality of the Arduino Duemilanove, but in a different package. It lacks only a DC power jack, and works with a Mini-B USB cable instead of a standard one.

Tech Specifications:

MICROCONTROLLER	ATmega328
ARCHITECTURE	AVR
OPERATING VOLTAGE	5 V
FLASH MEMORY	32 KB of which 2 KB used by bootloader
SRAM	2 KB
CLOCK SPEED	16 MHz
ANALOG IN PINS	8
EEPROM	1 KB
DC CURRENT PER I/O PINS	40 mA (I/O Pins)
INPUT VOLTAGE	7-12V
DIGITAL I/O PINS	22 (6 of which are PWM)
PWM OUTPUT	6
POWER CONSUMPTION	19 mA
PCB SIZE	18 x 45 mm
WEIGHT	7 g

PRODUCT CODE	A000005

Pinout Diagram:



Power:

The Arduino Nano can be powered via the Mini-B USB connection, 6-20V unregulated external power supply (pin 30), or 5V regulated external power supply (pin 27). The power source is automatically selected to the highest voltage source.

Memory:

The ATmega328 has 32 KB, (also with 2 KB used for the bootloader. The ATmega328 has 2 KB of SRAM and 1 KB of EEPROM.

Input and Output

Each of the 14 digital pins on the Nano 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. These pins are connected to the corresponding pins of the FTDI 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 Nano has 8 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 analogReference() function. Analog pins 6 and 7 cannot be used as digital pins. Additionally, some pins have specialized functionality:

- I2C: A4 (SDA) and A5 (SCL). Support I2C (TWI) communication using the Wire library (documentation on the Wiring website).

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 Nano has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provide UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An FTDI FT232RL on the board channels this serial communication over USB and the FTDI drivers (included with the Arduino software) provide a virtual com port to software on the computer. 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 FTDI chip and USB connection to the computer (but not for serial communication on pins 0 and 1). A Software Serial library allows for serial communication on any of the Nano'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. To use the SPI communication, please see ATmega328 datasheet.

Programming

The Arduino Nano can be programmed with the Arduino IDE. Select "Arduino Duemilanove or Nano w/ ATmega328" from the Tools > Board menu (according to the microcontroller on your board). The ATmega328 on the Arduino Nano 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. You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header using Arduino ISP or similar.

Automatic (Software) Reset

Rather than requiring a physical press of the reset button before an upload, the Arduino Nano 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 FT232RL is connected to the reset line of the ATmega328 via a 100 nano farad 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 Nano 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 Nano. 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.

● *Coding Involved:*

```
// Constants for ultrasonic sensor
const int trigPin = 4;
const int echoPin = 5;

// Constants for relay module
const int relay1Pin = 2;
const int relay2Pin = 3;

// Constants for water pump
const int pumpDuration = 5000; // Pump activation duration in milliseconds

void setup() {
  // Initialize the Serial monitor
  Serial.begin(9600);

  // Configure the relay pins as output
  pinMode(relay1Pin, OUTPUT);
  pinMode(relay2Pin, OUTPUT);

  // Configure the ultrasonic sensor pins
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
}

void loop() {
  // Read distance from the ultrasonic sensor
  float distance = getDistance();
```

```
  // Check if distance is within the activation range (2 cm to 12 cm)
  if (distance >= 2 && distance <= 12) {
    // Activate the water pump
    activatePump();
  } else {
    // Deactivate the water pump
    deactivatePump();
  }

  // Delay before the next reading
  delay(500);
}

float getDistance() {
  // Send a short ultrasonic pulse to trigger measurement
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // Read the echo duration
  long duration = pulseIn(echoPin, HIGH);

  // Convert the duration to distance (cm)
  float distance = duration * 0.034 / 2;

  // Print the distance on the Serial monitor
  Serial.print("Distance: ");
  Serial.print(distance);
  Serial.println(" cm");

  return distance;
```

```
}

void activatePump() {
  // Turn on both relay channels
  digitalWrite(relay1Pin, HIGH);
  digitalWrite(relay2Pin, HIGH);

  // Activate the pump for the specified duration
  delay(pumpDuration);

  // Turn off both relay channels
  digitalWrite(relay1Pin, LOW);
  digitalWrite(relay2Pin, LOW);
}

void deactivatePump() {
  // Turn off both relay channels
  digitalWrite(relay1Pin, LOW);
  digita// Constants for ultrasonic sensor
const int trigPin = 4;
const int echoPin = 5;

// Constants for relay module
const int relay1Pin = 2;
const int relay2Pin = 3;

// Constants for water pump
const int pumpDuration = 5000; // Pump activation duration in milliseconds

void setup() {
  // Initialize the Serial monitor
  Serial.begin(9600);

  // Configure the relay pins as output
```

```
  pinMode(relay1Pin, OUTPUT);
  pinMode(relay2Pin, OUTPUT);

  // Configure the ultrasonic sensor pins
  pinMode(trigPin, OUTPUT);
  pinMode(echoPin, INPUT);
}

void loop() {
  // Read distance from the ultrasonic sensor
  float distance = getDistance();

  // Check if distance is within the activation range (2 cm to 12 cm)
  if (distance >= 2 && distance <= 12) {
    // Activate the water pump
    activatePump();
  } else {
    // Deactivate the water pump
    deactivatePump();
  }

  // Delay before the next reading
  delay(500);
}

float getDistance() {
  // Send a short ultrasonic pulse to trigger measurement
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);

  // Read the echo duration
```

```
  long duration = pulseIn(echoPin, HIGH);

  // Convert the duration to distance (cm)
  float distance = duration * 0.034 / 2;

  // Print the distance on the Serial monitor
  Serial.print("Distance: ");
  Serial.print(distance);
  Serial.println(" cm");

  return distance;
}

void activatePump() {
  // Turn on both relay channels
  digitalWrite(relay1Pin, HIGH);
  digitalWrite(relay2Pin, HIGH);

  // Activate the pump for the specified duration
  delay(pumpDuration);

  // Turn off both relay channels
  digitalWrite(relay1Pin, LOW);
  digitalWrite(relay2Pin, LOW);
}

void deactivatePump() {
  // Turn off both relay channels
  digitalWrite(relay1Pin, LOW);
  digitalWrite(relay2Pin, LOW);
}lWrite(relay2Pin, LOW);
}
```

Truth Table for Relay Operation:

The water level in the tank	Relay operation (RL2)	Pump motor operation
Below low level	On	Starts
Above low level and below high level	On	Remains on
Reaches high level	Off	Stops

● *Connections*

1. Connect the Arduino Nano to your computer using a USB cable.
2. Power the Arduino Nano using a battery or USB power source.

For the dual-channel relay:

3. Identify the pins on the relay module labelled "IN1" and "IN2" for controlling each channel. Connect these pins to any available digital pins on the Arduino Nano (e.g., IN1 to digital pin 2 and IN2 to digital pin 3).
4. Connect the relay module's VCC and GND pins to the Arduino Nano's 5V and GND pins, respectively.
5. Connect the water pump to the relay module. Connect one terminal of the water pump to the COM (common) terminal of the relay, and connect the other terminal to the NO (normally open) terminal. Repeat this for both channels of the relay module.

For the ultrasonic sensor:

6. Connect the ultrasonic sensor's VCC and GND pins to the Arduino Nano's 5V and GND pins, respectively.

7. Connect the ultrasonic sensor's TRIG and ECHO pins to any available digital pins on the Arduino Nano (e.g., TRIG to digital pin 4 and ECHO to digital pin 5).

● Conclusion:

• In this busy day to day life to fill water tank is essential one for daily uses.

• In this case, some people may forget to switch on and switch off the motor properly.

• This may lead to wastage of electricity and water

• In order to address the problem, we must use this submersible pump automatic starter

• Through this starter, motor will turn on when there is insufficient water in tank and turn off automatically when reaches required limit in the tank

Reason behind No Submission:

We are very sorry to inform you that we are unable to submit our project work in running condition in this stipulated time period. You all know that we are getting very less time in this semester for the project. Besides this, some of our classmates had to join the respective companies where they are placed currently. And that's why we are only submitting the project report and explaining how this project is going to work actually.

● References

[1] S. M. Khaled Reza, Shah Ahsanuzzaman Md. Tariq, S.M. Mohsin Reza,"Microcontroller Based Automated Water Level Sensing and Controlling: Design and Implementation Issue", 0, San Francisco, USA

[2] Ria Sood, Manjit Kaur, Hemant Lenka , “ Design and Development of Automatic Water Flowmeter”, Mohali, India

[3] SanamPudasaini, Anuj Pathak, SukirtiDhakal, Milan Paudel,”Automatic Water Level Controller with Short Messaging Service (SMS) Notification”, Kathmandu University, Nepal.