TITLE

Pump Auto On - Off Control With

'Continuous Water Level Monitoring'

By Using Arduino Uno And Ultrasonic Sensor

GROUP MEMBERS

NAME	ROLL NUMBER
TANMOY SENGUPTA	10901617033
SWAGATAM KAR	10901617036
SUMAN SAURABH	10901617040
SUBHASIS SAMANTA	10901617043
VIDYA KUMARI	10901617031

Signature of Trainer

Signature of HOD

Signature of course coordinator

Objective

- Automatic turn ON pump when water level above 30 cm(from the sensor).
- Automatic turn OFF pump when water level less than 5 cm(from the sensor).
- Continuous display of water level in tank after a fixed interval.

PRE-REQUISITES
(KEY COMPONENTS USED)

ADRUINO UNO

The **Arduino Uno** is an open-source microcontroller board based on the Microchip ATmega328P microcontroller and developed by Arduino.cc .The board is equipped with sets of digital and analog input/output (I/O) pins that may be interfaced to various expansion boards (shields) and other circuits. The board has 14 digital I/O pins (six capable of PWM output), 6 analog I/O pins, and is programmable with the Arduino IDE (Integrated Development Environment), via a type B - USB cable. It can be powered by the USB cable or by an external 9-volt battery, though it accepts voltages between 7 and 20 volts. It is the most commercially used product, and also is similar to the **Arduino Nano** and **Leonardo**.

TECHNICAL SPECIFICATIONS :-

Operating Voltage:5 Volts

Input Voltage:7 to 20 Volts

Digital I/O Pins:14 (of which 6 can provide PWM output)

Analog Input Pins: 6

DC Current per I/O Pin: 20 mA

DC Current for 3.3V Pin:50 mA

Flash Memory:32 KB of which 0.5 KB used by boot loader

SRAM:2 KB

EEPROM:1 KB

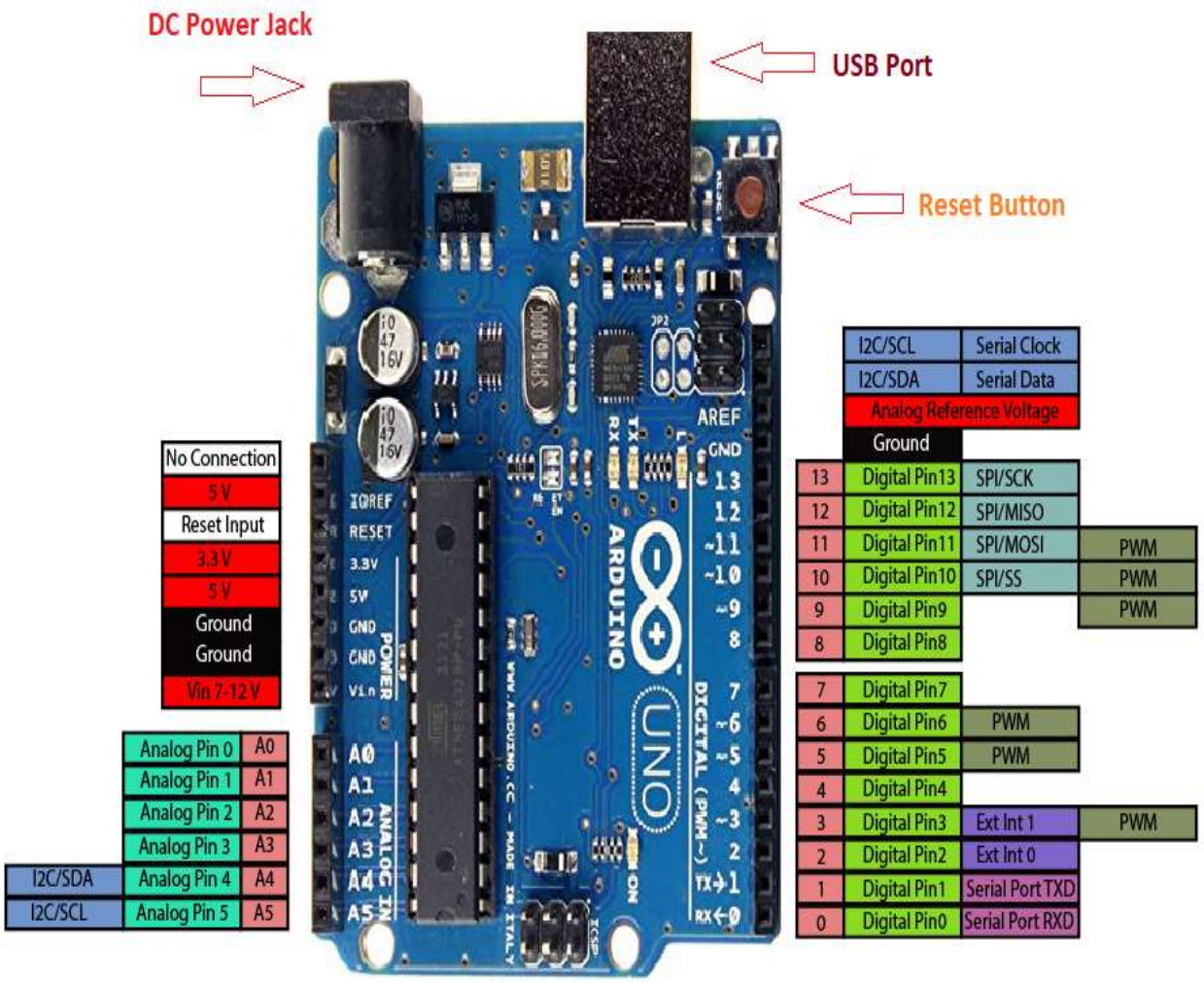
Clock Speed:16 MHz

Length: 68.6mm

Width:53.4 mm

Weight:25g

ARDUINO UNO PIN-OUT DETAILS



Special pin functions

Each of the 14 digital pins and 6 analog pins on the Uno can be used as an input or output, under software control (using pinMode(), digitalWrite(), and digitalRead() functions). They operate at 5 volts. Each pin can provide or receive 20 mA as the recommended operating condition and has an internal pull-up resistor (disconnected by default) of 20-50K ohm. A maximum of 40mA must not be exceeded on any I/O pin to avoid permanent damage to the microcontroller. The Uno has 6 analog inputs, labeled A0 through A5; each provides 10 bits of resolution (i.e. 1024 different values).

By default, they measure from ground to 5 volts, though it is possible to change the upper end of the range using the AREF pin and the analogReference() function. In addition, some pins have specialized functions:

Serial / UART: pins 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 ATmega8U2 USB-to-TTL serial chip.

External interrupts: pins 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.

PWM (pulse-width modulation): pins 3, 5, 6, 9, 10, and 11. Can provide 8-bit PWM output with the analogWrite() function.

SPI (Serial Peripheral Interface): pins 10 (SS), 11 (MOSI), 12 (MISO), and 13 (SCK). These pins support SPI communication using the SPI library.

TWI (two-wire interface) / I²C: pin SDA (A4) and pin SCL (A5). Support TWI communication using the Wire library.

AREF (analog reference): Reference voltage for the analog inputs.

Communication

The Arduino Uno has a number of facilities for communicating with a computer, another Arduino board, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The 16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, on Windows, a (.inf) file is required. Arduino Software (IDE) 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 USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1). A Software Serial library allows serial communication on any of the Uno's digital pins.

ULTASONIC SENSOR (HC-SR04)

The Ultrasonic Sensor HC-SR04 is one of the most commonly used distance measuring ultrasonic sensors and works extremely well with Arduino.

PINS

This module has 4 pins- VCC (5V), Trig, Echo, GND. Trig (trigger) is used to send out an ultrasonic high level pulse for at least 10µs and the Echo pin then automatically detects the returning pulse.

FIG- PINOUT DETAILS OF HC-SR04

TECHNICAL SPECIFICATIONS :-

Operating Voltage:5V Static current:2mA max Induction Angle:15° Detection Range:2 – 200cm High precision up to 3mm

MEASURING DISTANCES . . . .

The time it takes the sound wave to be sent, hit the object and return back to the sensor is measured. This time is then multiplied by the speed of sound (343m/sec = 0.0343cm/µs = [1/29.1] cm/µs approx.) to give the total distance traveled by the ultrasonic wave, which is then divided by 2 (to account for the fact that the wave was sent, hit the object, and then returned back to the sensor, hence covering twice the distance to the object) Distance = (Time for wave to return * Speed of sound) / 2.

This sensor can be used in any robot that needs to know the distances to objects in front of it. This sensor can be used with Chineduino Uno Rev3, Magician Robot Controller Board, Mini Driver Robot Controller or Spider Robot Controller .This sensor can also be mounted on a Pan Tilt Kit to detect objects in a wider area. A few example uses are:

1. Object Detecting Robot
2. Line Follower with Object Detecting Robot
3. Automatic water pump control operation(DONE IN THIS PROJECT)

RELAY MODULE (5V - DC)

The Single Relay Board can be used to turn lights, fans and other devices on/off while keeping them isolated from your microcontroller. The Single Relay Board allows you to control high-power devices (up to 10 A) via the on-board relay. Control of the relay is provided via a 1 x 3 header – friendly to servo cables and convenient to connect to many development boards.

TECHNICAL SPECIFICATIONS :-

Supply Voltage-5V

Control high-power devices up to 10 A with a simple high/low signal

Provides isolation between the microcontroller and device being controlled

Screw terminals for relay connections

3-pin servo-style header for power/signal interface

Equipped with a high-current relay (10A @ 28VDC)

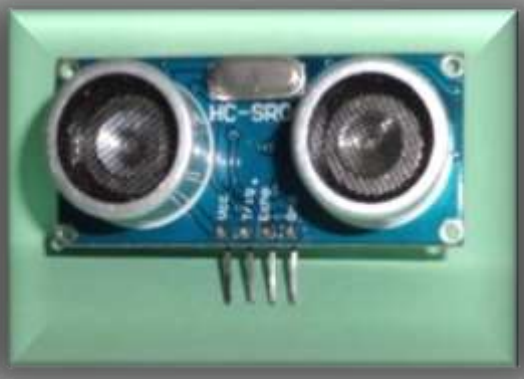
2x LED's that show the current state of the relay

COMPONENTS USED

- Single Channel Relay Module * (1 No.s)
- Arduino Uno * (1 No.s)
- Ultrasonic Sensor (HCSR04) * (1 No.s)
- LED Bulb * (1 No.s) [To simulate a pump]
- Bread board * (1 No.s)
- Jumper cables * (1 Lot)

DIAGRAMS

ARDUINO UNO



ULTRASONIC SENSOR

RELAY MODULE

Circuit Diagram

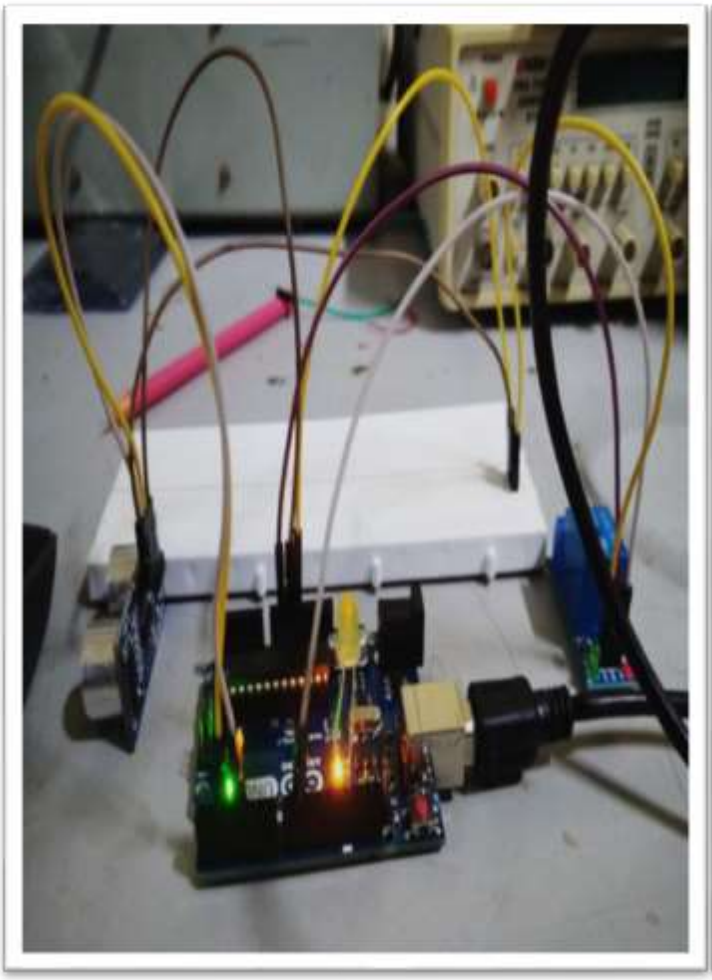


Fig-Water level between 5 cm and 30 cm(from the sensor – NO PUMP OPERATION)

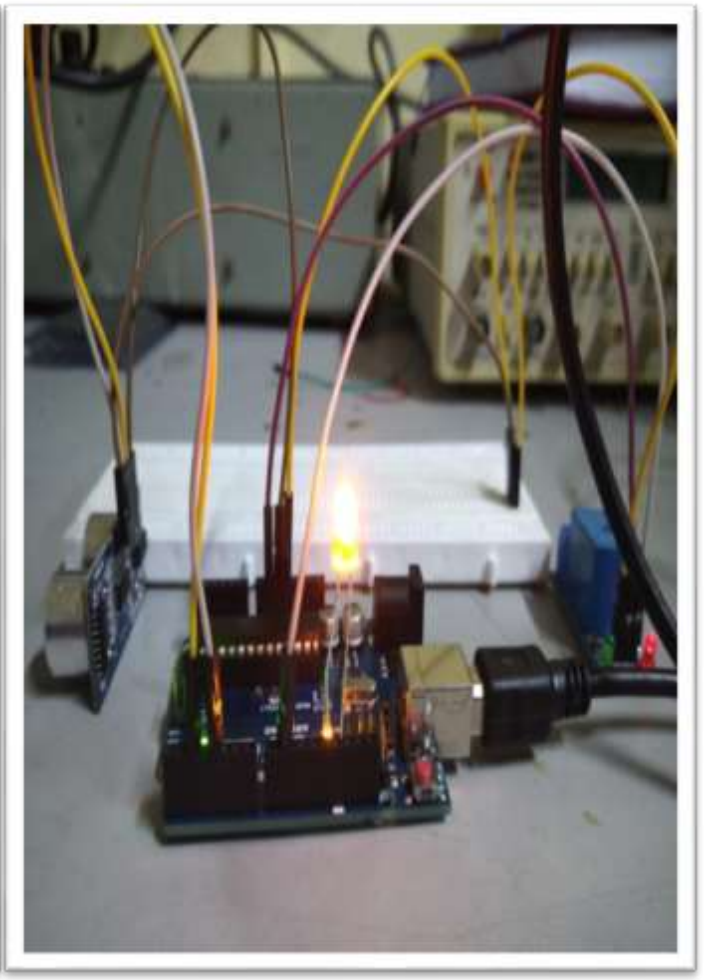


Fig-Water level <= 5 cm or >= 30 cm (LED HIGH to indicate HIGH LEVEL / LOW LEVEL)

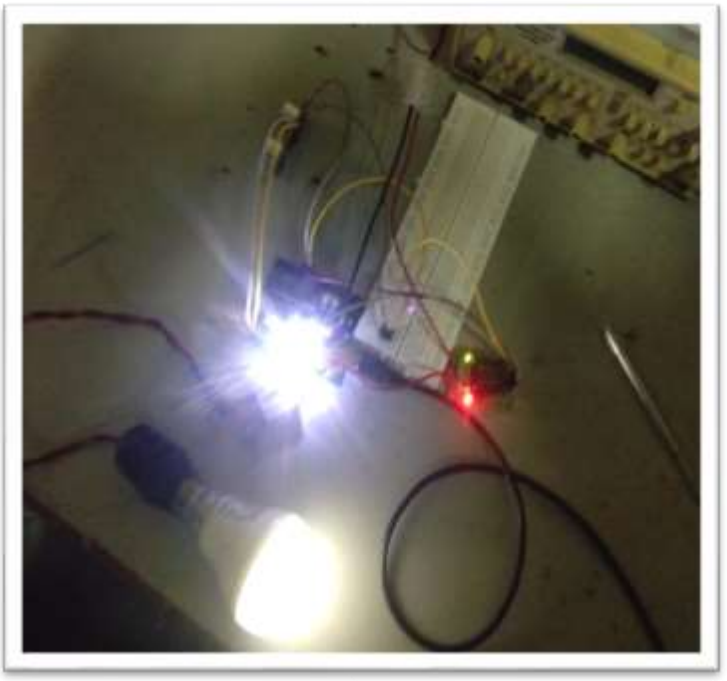


Fig- Bulb glowing (PUMP in operation, when water level >= 30 cm)

Connection Diagram

Arduino	HC-SR04
5v	Vcc
GND	GND
4(Digital)	Trig
2(Digital)	Echo

Fig-Pin Diagram For Arduino UNO and Ultrasonic Sensor HC-SR04

Arduino	Relay
5v	Vcc
GND	GND
9(Digital)	IN

Fig-Pin Diagram For Arduino UNO and Relay.

Programming Code

```
const int trigPin = 4;
const int echoPin = 2;
const int relay = 9;
const int led = 13;
long duration;
float level;

void setup()
{
  pinMode(trigPin, OUTPUT);              // Sets the trigPin as an Output
  pinMode(relay, OUTPUT);                // Sets the relay as an Output-----for turning pump ON and OFF
  pinMode(led, OUTPUT);                  // Sets the LED as an Output
  pinMode(echoPin, INPUT);               // Sets the echoPin as an Input
  digitalWrite (led, LOW);
  Serial.begin(9600);                    // Starts the serial communication
}

void loop()
{
  digitalWrite(trigPin, LOW);
  delay(200);
  digitalWrite(trigPin, HIGH);
  delay(1000);
  digitalWrite(trigPin, LOW);
  duration = pulseIn(echoPin, HIGH);
  float level = duration*0.034/2;
  delay (300);

  if (level <=5.0)
  {
    digitalWrite(led, HIGH);
```

```
  duration = pulseIn(echoPin, HIGH);
  float level = duration*0.034/2;
  delay (300);

  if (level <=5.0)
  {
    digitalWrite(led, HIGH);
    digitalWrite(relay, LOW);
    Serial.println("Water level HIGH, stop pump");
  }

  else if (level >=30.0)
  {
    digitalWrite(led, HIGH);
    digitalWrite(relay, HIGH);
    Serial.println("Water level LOW, start pump");
  }
  else
  {
    //BY Dsync Team
    digitalWrite(led,LOW);
    digitalWrite(relay, LOW);
  }
  Serial.print("Current water level in cm: ");
 Serial.println(level);
 delay(2000);

}
```

Fig-Programming code for Pump ON/OFF Operation

End Result

```
11:27:18.550 -> Current water level in cm: 6.17
11:27:22.060 -> Current water level in cm: 5.76
11:27:25.532 -> Current water level in cm: 5.29
11:27:29.041 -> Current water level in cm: 5.75
11:27:32.564 -> Current water level in cm: 5.80
11:27:36.050 -> Current water level in cm: 8.30
11:27:39.583 -> Current water level in cm: 10.73
11:27:43.088 -> Current water level in cm: 14.33
11:27:46.603 -> Current water level in cm: 16.86
11:27:50.090 -> Current water level in cm: 17.34
11:27:53.600 -> Current water level in cm: 22.42
11:27:57.097 -> Current water level in cm: 26.44
11:28:00.598 -> Current water level in cm: 28.56
11:28:04.097 -> Water level LOW, start pump
11:28:04.144 -> Current water level in cm: 30.46
11:28:07.600 -> Water level LOW, start pump
11:28:07.644 -> Current water level in cm: 31.06
11:28:11.135 -> Water level LOW, start pump
11:28:11.135 -> Current water level in cm: 30.65
11:28:14.655 -> Water level LOW, start pump
11:28:14.655 -> Current water level in cm: 30.63
11:28:18.162 -> Water level LOW, start pump
11:28:18.162 -> Current water level in cm: 30.69
11:28:21.651 -> Water level LOW, start pump
11:28:21.693 -> Current water level in cm: 30.65
11:28:25.138 -> Water level LOW, start pump
11:28:25.176 -> Current water level in cm: 30.70
11:28:28.657 -> Water level LOW, start pump
11:28:28.692 -> Current water level in cm: 30.70
11:28:32.187 -> Water level LOW, start pump
11:28:32.187 -> Current water level in cm: 30.69
11:28:35.659 -> Water level LOW, start pump
11:28:35.700 -> Current water level in cm: 30.75
11:28:39.161 -> Water level LOW, start pump
11:28:39.200 -> Current water level in cm: 30.74
11:28:42.698 -> Water level LOW, start pump
11:28:42.698 -> Current water level in cm: 30.75
11:28:46.212 -> Water level LOW, start pump
11:28:46.212 -> Current water level in cm: 31.16
```

```
11:28:49.688 -> Water level LOW, start pump
11:28:49.723 -> Current water level in cm: 30.80
11:28:53.203 -> Water level LOW, start pump
11:28:53.241 -> Current water level in cm: 30.80
11:28:56.709 -> Water level LOW, start pump
11:28:56.758 -> Current water level in cm: 30.79
11:29:00.223 -> Water level LOW, start pump
11:29:00.271 -> Current water level in cm: 30.80
11:29:03.719 -> Water level LOW, start pump
11:29:03.754 -> Current water level in cm: 30.80
11:29:07.224 -> Water level LOW, start pump
11:29:07.275 -> Current water level in cm: 30.80
11:29:10.740 -> Water level LOW, start pump
11:29:10.773 -> Current water level in cm: 30.80
11:29:14.255 -> Water level LOW, start pump
11:29:14.290 -> Current water level in cm: 30.80
11:29:17.770 -> Water level LOW, start pump
11:29:17.770 -> Current water level in cm: 30.80
11:29:21.244 -> Water level LOW, start pump
11:29:21.290 -> Current water level in cm: 30.80
11:29:24.753 -> Water level LOW, start pump
11:29:24.790 -> Current water level in cm: 30.09
11:29:28.300 -> Water level LOW, start pump
11:29:28.300 -> Current water level in cm: 30.26
11:29:31.783 -> Current water level in cm: 29.84
11:29:35.287 -> Water level LOW, start pump
11:29:35.339 -> Current water level in cm: 30.36
11:29:38.790 -> Current water level in cm: 29.84
11:29:42.320 -> Current water level in cm: 29.99
11:29:45.805 -> Current water level in cm: 26.33
11:29:49.314 -> Current water level in cm: 22.83
11:29:52.807 -> Current water level in cm: 22.37
11:29:56.346 -> Current water level in cm: 17.09
11:29:59.816 -> Current water level in cm: 12.82
11:30:03.346 -> Current water level in cm: 8.25
11:30:06.829 -> Water level HIGH, stop pump
11:30:06.871 -> Current water level in cm: 4.62
11:30:10.350 -> Water level HIGH, stop pump
11:30:10.384 -> Current water level in cm: 4.47
```

Fig-Serial Monitor Data For Pump ON/OFF System

Conclusion

Automation of the various components around us has been widely increased to reduce human intervention and save time. It is known that improper water management can have harmful effects on both the system and the environment. The main objective of this project is not only to reduce manual labour but also help save water in an efficient manner. Finally, a conclusion can be drawn , that this project can definitely be useful on a large scale basis due to its minimum requirement of man power and also the installation process being easier making it more compatible for everyone to use.

Thank You