

Sensor based Automated Hand Sanitizer

Divyashree K L

Lecturer ECE Govt Polytechnic Chintamani

Shylaja S N,

Lecturer Electronics and Communication Engineering Government Polytechnic for Women, Hassan

Introduction

Hygiene is an important aspect to remain healthy. There are various aspects of hygiene. A clean hand is one of them. Hands generally are touched at various surfaces and can be exposed to direct contamination. Cleaning hands at regular interval is recommended by various health organizations including WHO. Hand hygiene is now regarded as one of the most important element of infection control activities. In the wake of the growing burden of health care associated infections (HCAIs), the increasing severity of illness and complexity of treatment, superimposed by multi-drug resistant (MDR) pathogen infections, health care practitioners (HCPs) are reversing back to the basics of infection preventions by simple measures like hand hygiene. This is because enough scientific evidence supports the observation that if properly implemented, hand hygiene alone can significantly reduce the risk of cross-transmission of infection in healthcare facilities (HCFs)1-5. Evidence suggests that hand sanitization significantly reduces the transmission of healthcare-associated pathogens and the incidence of HCAI (healthcare associated infections). According to the Centre for Disease Control and Prevention (CDC), hand hygiene encompasses the cleansing of your hands using soap and water, antiseptic hand washes, alcohol-based hand sanitizers (ABHS), or surgical hand antiseptics. These days, alcohol-based hand sanitizers are increasingly being used instead of soap and water for hand hygiene in healthcare settings. Poor or inadequate hand washing and/or hand hygiene is known to be problematic in hospital settings, and is a major source of infections contracted while patients are admitted to a hospital. While hand washing and hygiene policies and training are important and can be effective in reducing the spread of infections, the problem of infections due to unsatisfactory hygiene of staff, medical professionals, and even patients continues to be problematic. It is known to place hand washing stations and hand sanitizer dispensers throughout medical facilities including in examination rooms, hallways, lobbies, and even patient rooms. However, such systems are purely mechanical and are incapable of providing an automated means of establishing accountability of good hygienic practices. During the last quarter of 2019, a collection of unusual pneumonia cases went from a local concern to a global pandemic in a matter of 70 days. The infamous Severe Acute Respiratory Syndrome Coronavirus 2

(SARS-CoV-2) is the virus [8] that was first reported in Wuhan, China on December 31, 2019, and was announced as a pandemic by the World Health Organization on March 11, 2020. This virus is zoonotic (a virus that is transmitted between animals and humans) and originates from bats. Besides, this virus can also be transmitted from humans to humans. Coronavirus can be transmitted either by air, direct contact, or indirectly. However, it is most commonly spread by droplets. Symptoms caused by this virus include the mild flu, namely a cold, sore throat, cough, fever, and difficulty breathing. In severe cases, Covid-19 can manifest as pneumonia. Patients can develop acute respiratory distress syndrome for a short time and die from multiple organ failure. The existence of this disease has a big impact on both socials and economics. WHO has declared this a pandemic disease and many cities around the world are in a lockdown situation. To prevent the cause of this virus, it can be done by keeping a distance at least 1 meter, avoid going to crowded places, avoid touching the eyes, mouth, and nose when outside, and cleaning hands with soap or alcohol-based hand rub. Providing containers for cleaning fluids in public spaces is a form of Covid-19 prevention, but the provision of containers is currently ineffective because there are parts that are often touched. This could be a point of transmission for Covid-19. Many health actions are carried out using automatic systems including air quality monitoring, hand sanitizers, hand hygiene. Hand sanitizers are an alternative for washing hands during a pandemic. It can be used when and water are not available. Hand sanitizer is also available in several forms such as liquid (spray) or gel. Hand sanitizer is usually made from materials such as alcohol, polyacrylic acid, glycerine, propylene glycol, or plant extracts. The process of killing germs starts with removing the oil on the skin, then the bacteria in the body will come to the surface. Soap or alcohol will kill bacteria after rubbing to your hand. Hand sanitizer is effective against

Covid-19. So far, most of the available hand sanitizers do not operate automatically. This article aims to make an automatic hand sanitizer where sanitizer liquid can come out automatically. Here, the circuit includes an ultrasonic sensor SC-04. The sensor senses the proximity of hands under the machine. The machine is designed for wall mount at a height of 4ft such that anyone can reach to get sanitizer dispense. The sensor send signal to the microcontroller and the controller takes decision to actuate the pump and valve simultaneously to dispense the liquid sanitizer through a mist nozzle.

Methodology

Hand sanitizer s need to be prepared in each classroom. Making an Automatic Hand sanitizer based on a Microcontroller is an automatic system that functions to increase the efficiency of using the hand sanitizer so that it is more effectively used and does not run out quickly in use. Automatic hand sanitizer is useful for making it easier for the hand sanitizer liquid to come out of the bottle, so that students don't have to press the bottle first. This Automatic Hand sanitizer uses the Arduino Leonardo Pro Micro as a microcontroller and an infrared sensor as a human hand detector. The pump motor gets power from a 5 volt battery and acts as an actuator that will press the bottle automatically. The mouth of the bottle is hand sanitizer connected using an elastic hose that leads to the part where the sanitizer liquid comes out. hand sanitizer This automatic was developed with the aim of improving strict health protocols, so that no Covid-clusters are found new19 in schools.

Proposed System

The proposed system consists of Arduino Uno, Ultrasonic Sensor, Water pump Motor, LEDs etc.

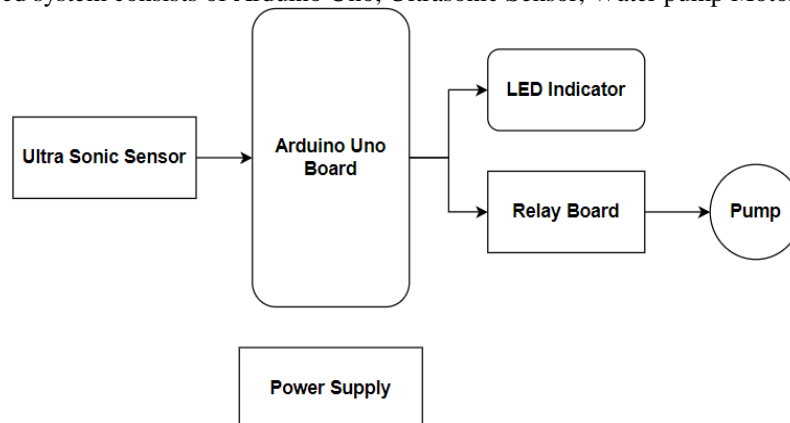


Fig: 2.1 Functioning Block Diagram

- **Arduino:** The Arduino is a microcontroller board based on the Atmega328. It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 16 analog inputs, 1 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 Arduino board is compatible with most shields designed for the Uno and the former boards Duemilanove or Diecimila.
- **Ultrasonic Sensor:** An ultrasonic sensor is an instrument that measures the distance to an object using ultrasonic sound waves. An ultrasonic sensor uses a transducer to send and receive ultrasonic pulses that relay back information about an object's proximity..
- **Water Pump:** We are designing our own vehicle unit which consists of 2 DC motor based wheels. These wheels are operated using 12v DC motor. The Microcontroller works at 5v and the DC motors operate at 12V, so to match the voltages we are interfacing a Relay which will in turn drive the DC motors.
- **Ultrasonic ranging module HC - SR04** provides 2cm - 400cm non-contact measurement function, the ranging accuracy can reach to 3mm. The modules includes ultrasonic transmitters, receiver and control circuit.

Hardware Requirements

The system is built using the following components.

- ✎ Arduino Microcontroller Board
- ✎ DC Power supply unit
- ✎ Ultrasonic Sensor SR-04
- ✎ Motor Pump
- ✎ Relay Board
- ✎ LEDs

3.1 Heart of the system: Microcontroller ARDUINO ARDUINO Uno:



Arduino is a popular programmable board used to create projects. It consists of a simple hardware platform as well as a free source code editor which has a “one click compile or upload” feature. Hence it is designed in way that one can use it without necessarily being an expert programmer (Kushner 1987). Arduino offers an open-source electronic prototyping platform that is easy to use and flexible for both the software and

hardware. Arduino is able to sense the environment through receiving input from several sensors. It is also able to control its surrounding through controlling motors, lights and other actuators.

The Arduino programming language that is based on the wiring and the Arduino development environment that is based on the processing are used to program the microcontroller found on the board (Banzi, 2005). Due to its open-source environment, one is able to easily write and upload codes to the I/O board. It is also worth to note that Arduino can be run on Linux, Mac OSX and Windows as its environment is written in Java

The Arduino Uno is a small, complete, and breadboard-friendly board based on the ATmega328 (Arduino Uno 3.x) or ATmega168 (Arduino Uno 2.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. The Uno was designed and is being produced by Gravitech.

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

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.

Features

Microcontroller	ATmega328
Operating Voltage	5V
Digital I/O Pins	14 (of which 6 provide PWM output)

Analog Input Pins	6
DC Current per I/O Pin	40 mA
Flash Memory	32 KB (ATmega328)
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz
Length	68.6 mm
Width	53.4 mm
Weight	25 g

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 ATmega168 has 16 KB of flash memory for storing code (of which 2 KB is used for the boot loader); the ATmega328 has 32 KB, (also with 2 KB used for the boot loader). The ATmega168 has 1 KB of SRAM and 512 bytes of EEPROM (which can be read and written with the [EEPROM library](#)); 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 [pin Mode\(\)](#), [digital Write\(\)](#), and [digital Read\(\)](#) 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 [attach Interrupt \(\)](#) function for details.
- PWM: 3, 5, 6, 9, 10, and 11. Provide 8-bit PWM output with the [analog Write\(\)](#) 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 it is possible to change the upper end of their range using the [analog Reference\(\)](#) function. Analog pins 6 and 7 cannot be used as digital pins. Additionally, some pins have specialized functionality:

- I²C: 4 (SDA) and 5 (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 [analog Reference \(\)](#).
- 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 ATmega168 ports.

Communication

The Arduino Nano has a number of facilities for communicating with a computer, another Arduino, or

other microcontrollers. The ATmega168 and 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 ATmega168 and 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 ATmega168 or ATmega328 datasheet.

Programming

The Arduino Nano can be programmed with the Arduino software ([download](#)). Select "Arduino Diecimila, Duemilanove, or Nano w/ ATmega168" or "Arduino Duemilanove or Nano w/ ATmega328" from the Tools > Board menu (according to the microcontroller on your board). For details, see the [reference](#) and [tutorials](#).

The ATmega168 or 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 ([reference](#), [C header files](#)).

You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header using [Arduino ISP](#) or similar; see [these instructions](#) for details.

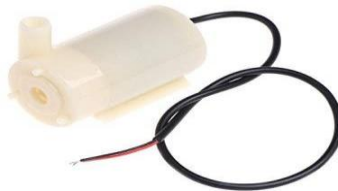
Automatic (Software) Reset

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 ATmega168 or 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 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.

Motor Pump:

The 12V DC Geared Motor can be used in variety of robotics applications and is available with wide range of RPM and Torque.



Description:

The 5V DC Mini Motor Pump can be used in variety of liquid flow applications and is available with wide range of RPM and water flow rate.

Features

- Suitable for use in aquariums, keeping and rearing tanks or small water features
- Where reliable water conveyance, circulation or flow is required.
- This water pump can be submersible, which it is convenient and useful.
- Flow rate can up to 100L/H. Current: 100-200mA, Flow Rate Max: 100L/H
- Mini but silent.

Specification:

- Material: Plastic
- Type: DC3V-5V/Submersible
- Current: 100-200mA
- Flow Rate Max: 100L/H
- H. Max : 0.8m
- Size: One Size
- Outlet: Outside
- Diameter: 7.5mm(0.3in)
- Inside Diameter: 4.5mm(0.18in)

3.2 Relay

Relay is an electromagnetic device which is used to isolate two circuits electrically and connect them magnetically. They are very useful devices and allow one circuit to switch another one while they are completely separate. They are often used to interface an electronic circuit (working at a low voltage) to an electrical circuit which works at very high voltage. For example, a relay can make a 5V DC battery circuit to switch a 230V AC mains circuit. Thus a small sensor circuit can drive, say, a fan or an electric bulb.



Relay

A **relay switch** can be divided into two parts: input and output. The input section has a coil which generates magnetic field when a small voltage from an electronic circuit is applied to it. This voltage is called the operating voltage. Commonly used relays are available in different configuration of operating voltages like 6V, 9V, 12V, 24V etc. The output section consists of contactors which connect or disconnect mechanically. In a basic relay there are three contactors: normally open (NO), normally closed (NC) and common (COM). At no input state, the COM is connected to NC. When the operating voltage is applied the relay coil gets energized and the COM changes contact to NO. Different relay configurations are available like SPST, SPDT, DPDT etc, which have different number of changeover contacts. By using proper combination of contactors, the electrical circuit can be switched on and off. Get inner details about structure of a relay switch.



3.1 Ultra Sonic Sensor (SR-04):

Ultrasonic ranging module HC - SR04 provides 2cm - 400cm non-contact measurement function, the ranging accuracy can reach to 3mm. The modules includes ultrasonic transmitters, receiver and control circuit.

The basic principle of work:

- (1). Using IO trigger for at least 10us high level signal,
- (2). The Module automatically sends eight 40 kHz and detect whether there is a pulse signal back.
- (3). IF the signal back, through high level , time of high output IO duration is the time from sending ultrasonic to returning. Test distance = (high level time×velocity of sound (340M/S) / 2,

Wire connecting direct as following:

- 5V Supply
- Trigger Pulse Input
- Echo Pulse Output
- 0V Ground

Electric Parameter:

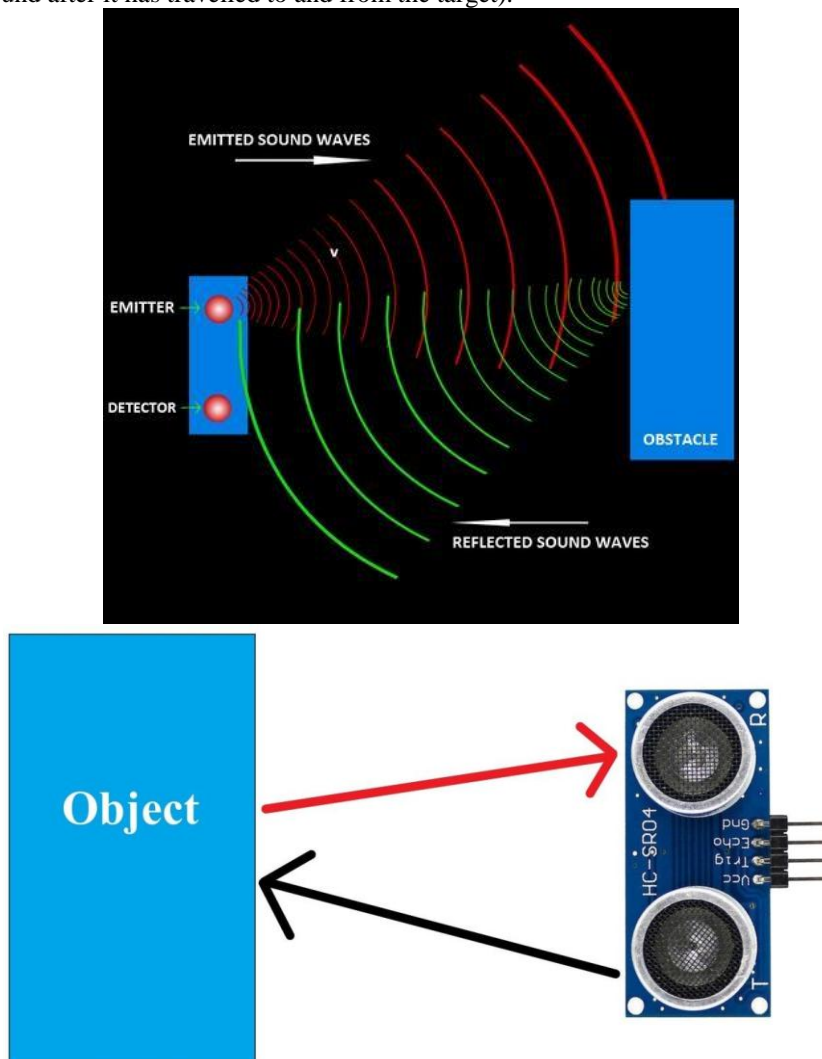
Working Voltage	DC 5 V
Working Current	15mA
Working Frequency	40Hz
Max Range	4m
Min Range	2cm
MeasuringAngle	15 degree
Trigger Input Signal	10uS TTL pulse
Echo Output Signal	Input TTL lever signal and the range in proportion
Dimension	45*20*15mm



- Pin 1 - VCC**
- Pin 2 - Trigger Pin**
- Pin 3 - Echo Pin**
- Pin 4 - GND**

3.2 Working of Ultrasonic Sensor

An ultrasonic sensor is an electronic device that measures the distance of a target object by emitting ultrasonic sound waves, and converts the reflected sound into an electrical signal. Ultrasonic waves travel faster than the speed of audible sound (i.e. the sound that humans can hear). Ultrasonic sensors have two main components: the transmitter (which emits the sound using piezoelectric crystals) and the receiver (which encounters the sound after it has travelled to and from the target).



In order to calculate the distance between the sensor and the object, the sensor measures the time it takes between the emission of the sound by the transmitter to its contact with the receiver.

The formula for this calculation is $D = \frac{1}{2} T \times C$

Where D is the distance, T is the time, and C is the speed of sound (~ 343 meters/second). For example, if a scientist set up an ultrasonic sensor aimed at a box and it took 0.025 seconds for the sound to bounce back, the distance between the ultrasonic sensor and the box would be: $D = 0.5 \times 0.025 \times 343$ or about **4.2875 meters**.

Ultrasonic sensors are used primarily as proximity sensors. They can be found in automobile self-parking technology and anti-collision safety systems. Ultrasonic sensors are also used in robotic obstacle detection systems, as well as manufacturing technology. In comparison to infrared (IR) sensors in proximity sensing applications, ultrasonic sensors are not as susceptible to interference of smoke, gas, and other airborne particles

(though the physical components are still affected by variables such as heat).

Ultrasonic sensors are also used as level sensors to detect, monitor, and regulate liquid levels in closed containers (such as vats in chemical factories). Most notably, ultrasonic technology has enabled the medical industry to produce images of internal organs, identify tumors, and ensure the health of babies in the womb.

3.5 12v Regulated Power Supply

The microcontroller and other devices get power supply from AC to DC adapter through 7805, 5 volts regulator. The adapter output voltage will be 12V DC non-regulated. The 7805/7812 voltage regulators are used to convert 12 V to 5V/12V DC.

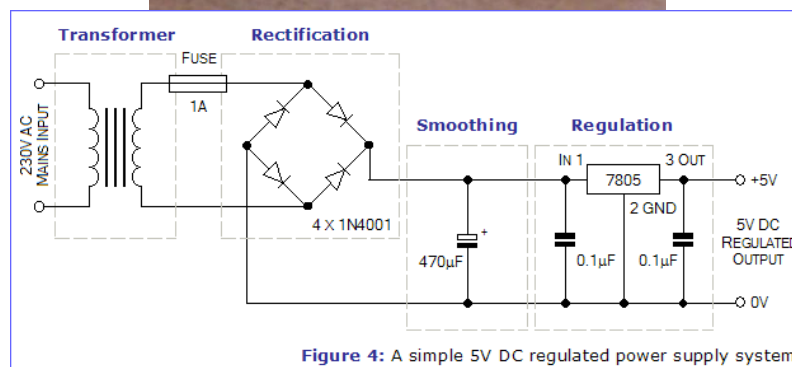


Fig 3.5: Power supply kit and circuit diagram

As shown in above figure, 230v 50Hz AC is applied to the step down 12-0-12 transformer which steps down input 230 volts into +/-12v and 1Amps. This voltage fed into diode rectifier circuit has four diodes are connected as bridge. It conducts a current in both half cycles of the input. I.e. two diodes conduct in two different half cycles. And produces alternating DC current of same range. Our design is to get a pure DC voltage required to power up the controller circuit. This is achieved using smoothing circuit consist of 470µF capacitor gives charging and discharging current which is almost equal to dc. Output of the smoothing circuit is fed into Lm7805 regulator which gives us a almost 5 volt DC current. Any Dc ripples are removed by .1µF capacitor. Finally output is of +5v DC suitable for microcontroller applications.

Software Requirements

The hardware components which are used are programmed by means of the software tools. The various software which are included are presented briefly.

Arduino IDE:

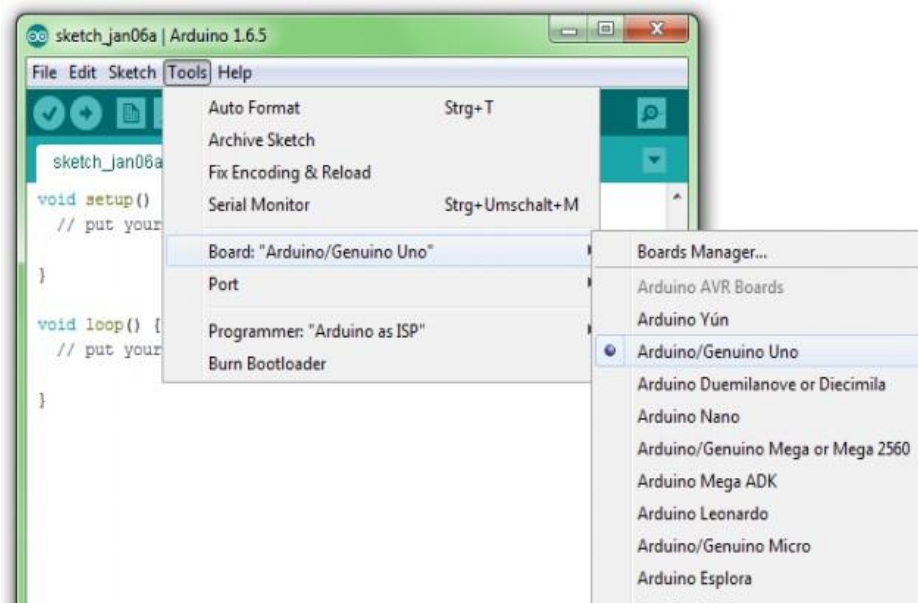
A program for ESP32 hardware may be written in any programming language with compilers that produce binary machine code for the target processor. Atmel provides a development environment for their 8-bit AVR

and 32-bit ARM Cortex-M based microcontrollers: AVR Studio (older) and Atmel Studio (newer)

The Arduino integrated development environment (IDE) is a cross-platform application (for Windows, macOS, Linux) that is written in the programming language Java. It originated from the IDE for the languages *Processing* and *Wiring*. It includes a code editor with features such as text cutting and pasting, searching and replacing text, automatic indenting, brace matching, and syntax highlighting, and provides simple *one-click* mechanisms to compile and upload programs to an ESP32board. It also contains a message area, a text console, a toolbar with buttons for common functions and a hierarchy of operation menus. The source code for the IDE is released under the GNU General Public License, version 2.^[56]

The ESP32IDE supports the languages C and C++ using special rules of code structuring. The ESP32IDE supplies a software library from the Wiring project, which provides many common input and output procedures. User-written code only requires two basic functions, for starting the sketch and the main program loop, that are compiled and linked with a program stub *main()* into an executable cyclic executive program with the GNU toolchain, also included with the IDE distribution. The ESP32IDE employs the program *avrdude* to convert the executable code into a text file in hexadecimal encoding that is loaded into the ESP32board by a loader program in the board's firmware.

2.2.1 Installation Now one after another the ESP32software and the USB driver for the board have to be installed. 2.2.1.1 Installation and set up of the ESP32 software 1. Download the ESP32software on www.arduino.cc and install it on the computer (The microcontroller NOT connected to the PC). After that you open the software file and start the program named *arduino.exe*. Two set ups on the program are important and should be considered. a) The board that you want to connect, has to be selected on the ESP32software. The "Funduino" is here known as "ESP32/ Genuino".

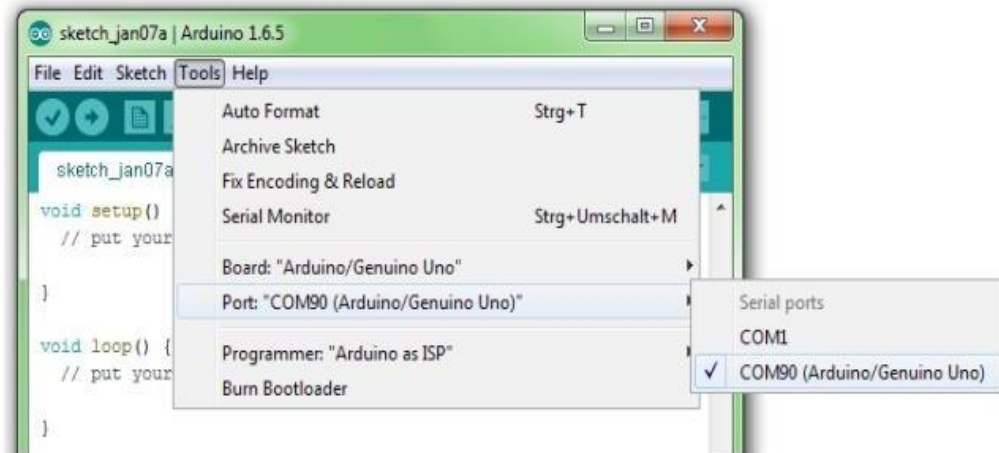


b) You have to choose the right "Serial-Port", to let the Computer know to which port the board has been connected. That is only possible if the USB driver has been installed correctly. It can be checked this way:

At the moment the ESP32 isn't connected to the PC. If you now choose "Port", under the field "Tool", you will already see one or more ports here (COM1/ COM2/ COM3...). The quantity of the shown ports doesn't depend on the quantity of the USB ports on the computer. When the board gets connected to the computer, YOU WILL FIND ONE MORE PORT.

2.2.1.2 Installation of the USB driver How it should be:

1. You connect the board to the computer.



2. The Computer recognizes the board and suggests to install a driver automatically.

Attention: Wait a second! Most of the time the computer can't find the driver automatically to install it. You might choose the driver by your own to install it. It can be found in the ESP32 file under "Drivers".

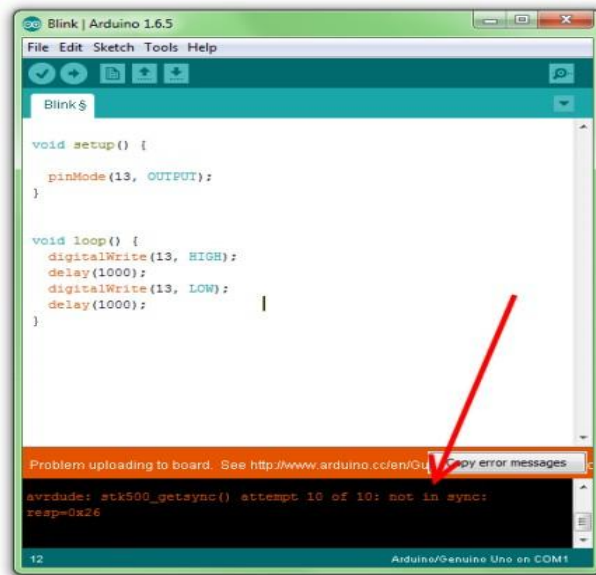
Control: At the control panel of the Computer you can find the "Device manager". If the board has been installed successfully, it should appear here. When the installation has failed, there is either nothing special to find or you will find an unknown USB device with a yellow exclamation mark. In this case: Click on the unknown device and choose "update USB driver". Now you can start over with the manual installation.

4.1 Programming

Now we can start properly. Without too much theoretical information we start directly with programming Learning by doing. On the left side you can find the "sketches", on the right the accompanying explanation for the commands in grey. If you work through the tutorials with this system, you will soon understand the code and be able to use it by yourself. Later on you can familiarize yourself with other features. These tutorials are only meant as first steps to the ESP32 world. All possible program features and codes are referred on www.arduino.cc under reference".

First of all a short explanation for possible error reports that can appear while working with the ESP32 software. The two most common ones are:

1) The board is not installed right or the wrong board is selected. After uploading the sketch, there will appear an error report underneath the sketch. It looks like the one in the picture on the right. The note "not in sync" shows up in the error report.



2.) There is a mistake in the sketch. For example, a word is misspelled or a bracket is missing. In the example on the left the last semicolon in the sketch is missing. In this Case the error report often starts with “excepted..”. This means that the program is still expecting something that is missing.



Basic structure of a sketch:

A sketch can be divided in three parts.

1. Name variable

In the first part elements of the program are named (This will be explained in program no. 3). This part is not absolutely necessary.

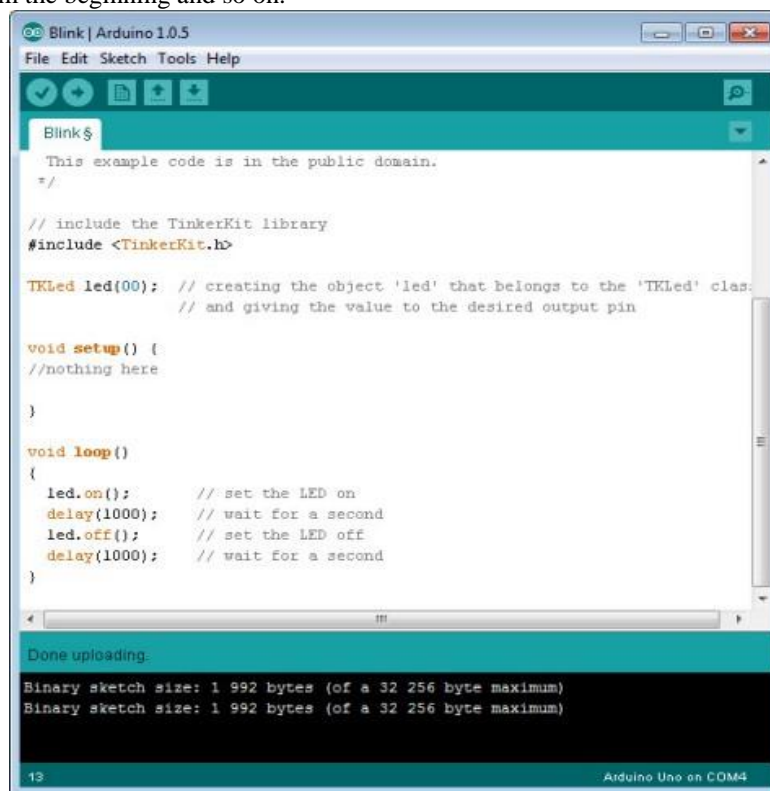
2. Setup (absolutely necessary for the program)

The setup will be performed only once. Here you are telling the program for example what Pin (slot for cables) should be an input and what should be an output on the boards.

Defined as Output the pin should put out a voltage. For example: With this pin a LED is meant to light up. Defined as an Input the board should read out a voltage. For example: A switch is actuated. The board recognized this, because it gets a voltage on the Input pin.

3. Loop (absolutely necessary for the program)

This loop part will be continuously repeated by the board. It assimilates the sketch from Beginning to end and starts again from the beginning and so on.



```
Blink | Arduino 1.0.5
File Edit Sketch Tools Help
Blink $
This example code is in the public domain.
*/

// include the TinkerKit library
#include <TinkerKit.h>

TKLed led(00); // creating the object 'led' that belongs to the 'TKLed' class
               // and giving the value to the desired output pin

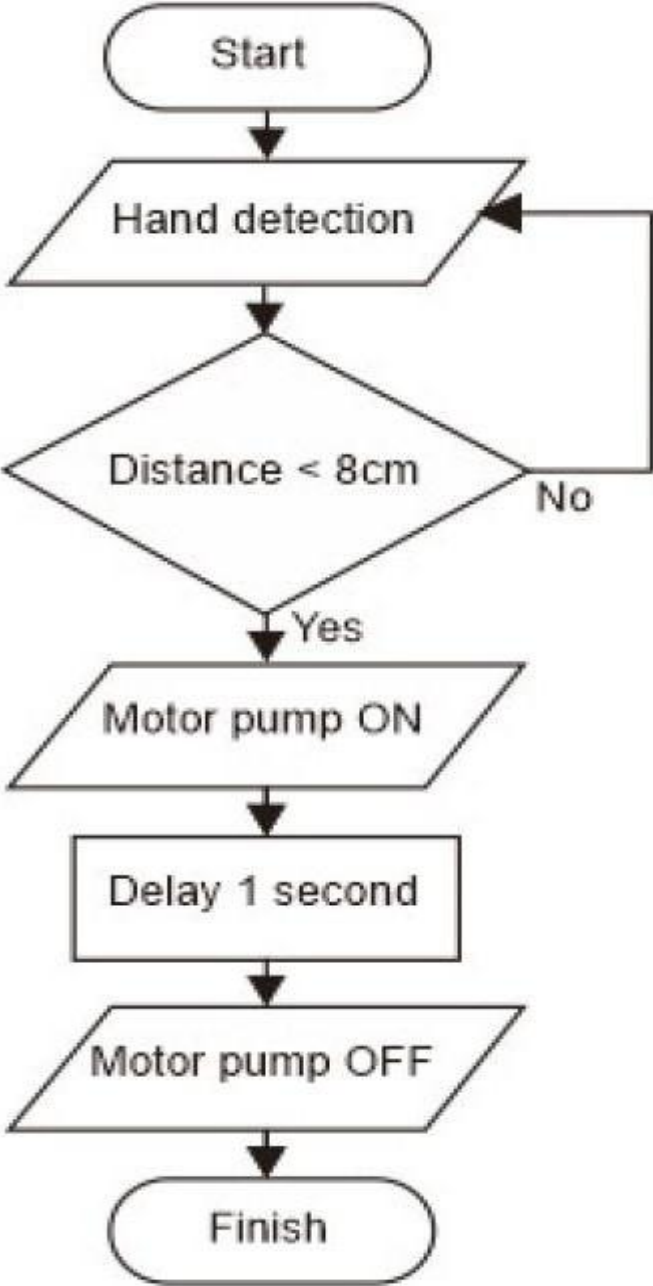
void setup() {
  //nothing here
}

void loop()
{
  led.on(); // set the LED on
  delay(1000); // wait for a second
  led.off(); // set the LED off
  delay(1000); // wait for a second
}

Done uploading.
Binary sketch size: 1 992 bytes (of a 32 256 byte maximum)
Binary sketch size: 1 992 bytes (of a 32 256 byte maximum)

13 Arduino Uno on COM4
```

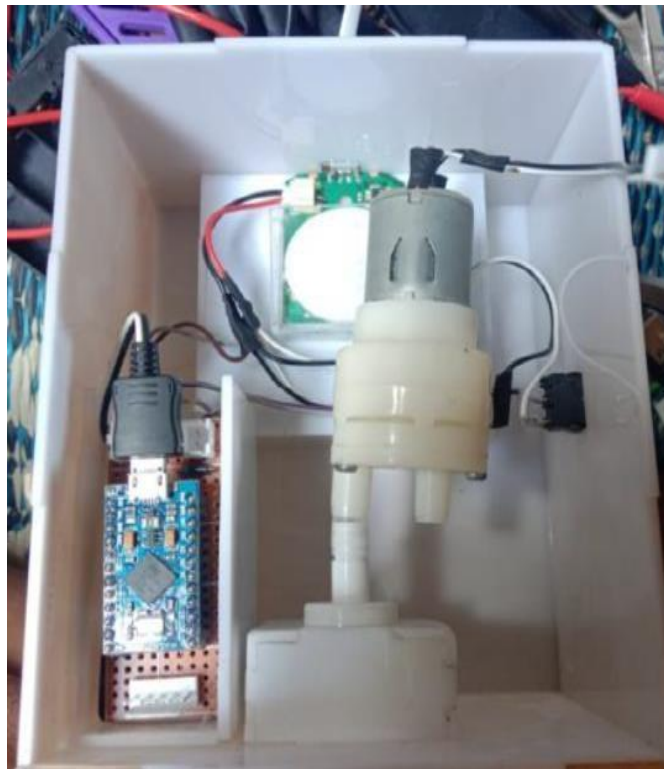
4.2 Project Flow



4.3 Project Code

```
#include <LiquidCrystal.h> #include <EEPROM.h>
LiquidCrystal lcd(7,6,5,4,3,2);
int led=13;
int start_addr = 0; int mem_addr = 80;
int temp=0,i=0,x=0,k=0; char str[100],msg[32];
void setup()
{
  lcd.begin(16,2); Serial.begin(9600); pinMode(led, OUTPUT); digitalWrite(led, HIGH);
  Welcome_message(); lcd.clear(); lcd.print("Initialzng..."); delay(2000);
  lcd.clear();
  lcd.print("GPT, Chintamani"); delay(1000);
  // lcd.clear();
  // lcd.setCursor(0,1);
  // lcd.print("System Ready");
  // Serial.println("AT+CNMI=2,2,0,0,0");
  // delay(500);
  // Serial.println("AT+CMGF=1");
  // delay(1000); digitalWrite(led, LOW);
}
void loop()
{
  for(unsigned int t=0;t<60000;t++)
  {
    for(unsigned int p=0;p<2;p++)
    {
      serialEvent(); if(temp==1)
      {
        x=0,k=0,temp=0; while(x<i)
        {
          while(str[x]!='#')
          {
            x++;
            while(str[x]!='*')
            {
              msg[k++]=str[x++];
            }
          }
          x++;
        }
        msg[k]='\0';
        lcd.clear(); lcd.print(msg); delay(1000);
        temp=0; i=0; x=0; k=0;
      }
    }
  }
  lcd.scrollDisplayLeft();
}
void serialEvent()
{
  while(Serial.available())
  {
    char ch=(char)Serial.read(); str[i++]=ch;
    if(ch == '*')
```

```
{  
temp=1; lcd.clear();  
lcd.print("Message Received"); delay(1000);  
}  
}  
}  
void Welcome_message()  
{  
lcd.clear();  
lcd.print(" Welcome to "); delay(1000);  
lcd.clear(); lcd.print(" Project on ");  
delay(1000); lcd.clear(); lcd.setCursor(0,0);  
lcd.print(" PC Controlled "); lcd.setCursor(0,1); lcd.print(" Notice Board "); delay(3000);  
}
```



Conclusion

Based on the results of the research on the design of the automatic hand sanitizer that the researchers did, it can be concluded that the hand sanitizer can work well when the hands are at a distance of 7 cm. According to the researchers, 7 cm is considered ideal because it has been adjusted to the discharge pipe for the hand sanitizer. The hand sanitizer can be active for approximately 20 hours and one time filling of the hand sanitizer liquid can be used up to 400 times.

References

- [1] Wu, Zunyou, and Jennifer M. McGoogan. "Characteristics of and important lessons from the coronavirus disease 2019 (COVID-19) outbreak in China: summary of a report of 72 314 cases from the Chinese Center for Disease Control and Prevention." *Jama* 323.13 (2020): 1239-1242.
- [2] Surat Edaran No. 4 Tahun 2020 tentang Pelaksanaan Kebijakan Pendidikan dalam Masa Darurat Penyebaran Corona Virus Disease (Covid-19).
- [3] World Health Organization. Protocol for assessment of potential risk factors for coronavirus disease 2019 (COVID-19) among health workers in a health care setting, 23 March 2020. No. WHO/2019-nCoV/HCW_risk_factors_protocol/2020.3. World Health Organization, 2020.
- [4] Wu, Huai-liang, et al. "Facemask shortage and the novel coronavirus disease (COVID-19) outbreak: Reflections on public health measures." *EclinicalMedicine* (2020): 100329.
- [5] Cheng, Kar Keung, Tai Hing Lam, and Chi Chiu Leung. "Wearing face masks in the community during the COVID-19 pandemic: altruism and solidarity." *The Lancet* (2020).
- [6] Saadat, S., Rawtani, D., & Hussain, C. M. (2020). Environmental perspective of COVID-19. *Science of The Total Environment*, 138870. doi:10.1016/j.scitotenv.2020.138870
- [7] Haryati, Sri. "Research and Development (R&D) sebagai salah satu model penelitian dalam bidang pendidikan." *Majalah Ilmiah Dinamika* 37.1 (2012): 15.
- [8] Gowda, Manushri, et al. "Power Consumption Optimization in IoT based Wireless Sensor Node Using ESP8266." *ITM Web of Conferences*. Vol. 32. EDP Sciences, 2020.
- [9] Tsauqi, Angga Khalifah, et al. "Saklar Otomatis Berbasis Light Dependent Resistor (Ldr) Pada Mikrokontroler Arduino Uno." *PROSIDING SEMINAR NASIONAL FISIKA (E-JOURNAL)*. Vol. 5. 2016.
- [10] Zhmud, V. A., et al. "Application of ultrasonic sensor for measuring distances in robotics." *Journal of Physics: Conference Series*. Vol. 1015. No. 3. 2018.
- [11] Badamasi, Yusuf Abdullahi, "The working principle of an Arduino." 2014 11th international conference on electronics, computer and computation (ICECCO). IEEE, 2014.
- [12] El Anwar, Yogie, Noer Soedjarwanto, and Ageng Sadnowo Repelianto. "Prototype penggerak pintu pagar otomatis berbasis arduino uno Atmega 328p