AMERICAN Journal of Engineering, Mechanics and Architecture
Volume 3, Issue 3, 2025 ISSN (E): 2993-2637

Auto FILL Fluid using Arduino

Alyaa Najeh Shamkhi Jabur, Aya Maged Adrenel Mansor, Ali Abd Alhaseeb Abd Alhusain Yahi
Al-Salam University, Medical Devices Engineering, Iraq

Ali Musheb Saud Hamad, Mohammed Salim Dawod Salih
Al-Hadi University College, Medical Devices Engineering, Iraq

Abstract: The research presents an automated bottle-filling system through a control system, where the system contains the main control unit, which is the Arduino, through which, and through its programming, we control all the instructions in light of the programming and the servo motor through which the mechanical part that is mounted on it rotates and will contain the bottles. It also contains a small keyboard through which any instructions can be given for the purpose of filling bottles. It also contains a water pump through which we perform the automatic flow of water as needed. It also contains a relay that operates the water pump and is equipped with a power of 5-7 volts.

This process can be used for health prevention without direct human intervention to prevent transmissible diseases that can be in the form of viruses or bacteria concentrated on the surfaces of objects. We can also develop this work and have it via the Internet, which we will talk about in the proposals in the coming chapters to develop the project.

Keywords: Arduino. Control system. Water pump. Human free operation. Servo motor. Keypad.

1. Introduction:

Automation has become a fundamental aspect of our everyday existence, influencing areas ranging from industrial production to home automation. With technological advancements, machines have evolved to become more intelligent and capable of executing complex tasks. This evolution has enabled higher levels of accuracy, productivity, and efficiency, while simultaneously reducing human error and lessening the need for manual labor. Microcontrollers, such as Arduino, have significantly contributed to the evolution of automation systems. These compact integrated devices provide a cost-effective alternative to more expensive programmable logic controllers (PLCs). Arduino can be programmed to manage a diverse array of devices and sensors, making it a versatile platform for various automation projects. The advantages of automation are extensive. By minimizing human involvement, automation can lower labor costs, enhance productivity, and improve safety in hazardous environments. Additionally, it facilitates consistent and repeatable outcomes, which can lead to improved product quality and greater customer satisfaction. However, automation also poses certain challenges. The initial investment required for automation implementation can be substantial, and ongoing maintenance and repair expenses may also be considerable. Moreover, automation can lead to job displacement, resulting in adverse social and economic effects. Indeed, the utilization of Programmable Logic Controllers (PLCs) for bottle filling operations in large-scale industries has been a standard practice for many years. PLCs are known for their high reliability and precision, making them an ideal choice for critical industrial applications such as the filling of soft drinks and pharmaceuticals. However, for small-scale manufacturers, the initial and operational costs associated with PLCs can be prohibitive. The utilization of Arduino microcontrollers presents a compelling alternative for automating bottle filling operations. Arduino provides a budget-friendly means to enhance automation while ensuring precision and dependability. It can oversee the complete bottle filling procedure, which includes identifying the presence of bottles, regulating the filling volume, and halting the operation once the target level is achieved. By incorporating sensors like ultrasonic and flow sensors, Arduino can effectively ascertain the position of the bottle and the filling level, making necessary adjustments to the filling process. Additionally, it can calculate the filling duration and implement a time delay for the pump, allowing it to cease operation once the glass is filled.

2. Methods:

The creation of an automated liquid filling system utilizing an Arduino UNO controller highlights the considerable benefits of automation in industrial processes. This system incorporates an electromechanical assembly that utilizes a disk or plate mechanism. A motor facilitates the rotation of this assembly, while a water pump is responsible for transferring liquid from a storage tank into the bottles. The Arduino UNO acts as the primary component overseeing the system's operations. Start and stop buttons are linked to the Arduino development board, which is programmed to control the motor and water pump based on sensor inputs. This setup reduces the necessity for human involvement and decreases the likelihood of errors, resulting in a highly reliable and efficient filling process. Initially, the automatic liquid filling system is filled with liquid, and the operator is instructed to initiate the process by pressing the "START" button on the Arduino UNO controller. Following this, the system activates the water pump to commence the filling operation. Bottles are transported along a conveyor belt equipped with sensors that track their position and movement. The Arduino UNO controller employs these sensor readings to adjust the liquid filling for each bottle, ensuring that the process is carried out with accuracy and efficiency, thus minimizing waste and spillage.

3.Project Components:

3.1. Arduino UNO:

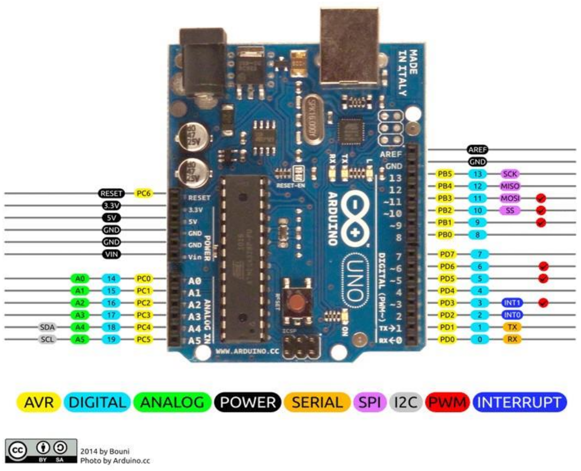

Fig. (1): Arduino UNO R3

The Arduino Uno is a microcontroller board that features the ATmega328 (see the datasheet for details). It includes 14 digital input/output pins, of which 6 are capable of functioning as PWM outputs, in addition to 6 analog input pins. The board is equipped with a 16 MHz ceramic resonator, a USB interface, a power jack, an ICSP header, and a reset button. It contains all essential components to support the microcontroller, allowing users to connect it to a computer via a USB cable or power it using an AC-to-DC adapter or battery to get started. Unlike earlier models, the Uno does not include the FTDI USB-to-serial driver chip; instead, it utilizes the Atmega16U2 (or Atmega8U2 in versions up to R2), which is programmed to function as a USB-to-serial converter. The second revision of the Uno board features a resistor that connects the 8U2 HWB line to ground, simplifying the process of entering DFU mode. The 1.0 pinout introduces SDA and SCL pins located near the AREF pin, as well as two additional pins near the RESET pin: the IOREF pin, which allows shields to adapt to the voltage supplied by the board. In the future, shields will be compatible with both boards that use the AVR, operating at 5V, and the Arduino Due, which operates at 3.3V. The second pin is currently unconnected and reserved for future use.

3.2. Servo Motor:

Fig (2): Servo motor 360

The Servo Motor is an electric motor equipped with a sophisticated control system that enables it to rotate to designated angles with precision. It functions through a PWM (Pulse Width Modulation) signal, which dictates its exact positioning. The assembly includes either a DC or AC motor, a gearbox, and an electronic control circuit. Renowned for its rapid response and exceptional accuracy, the servo motor is particularly suited for applications that demand precise angle control, such as in robotics, 3D printing, and drone technology. Typically, the servo motor has a restricted rotation range, generally spanning from 0° to 180°. It can be operated using microcontrollers such as Arduino or Raspberry Pi. Its applications are extensive, encompassing industrial automation, medical equipment, and remote-control systems. Unlike conventional motors, the servo motor does not provide continuous rotation; instead, it moves to a specific position and maintains that position.

3.2.1. What makes a continuous servo motor special?

A standard servo motor requires a PWM signal of approximately 50Hz, with pulse widths varying between 1ms and 2ms to regulate the position of its shaft. For instance, when the pulse width is set to 1.5ms, the motor will promptly rotate to a 90-degree position and hold that angle until the pulse width is adjusted.

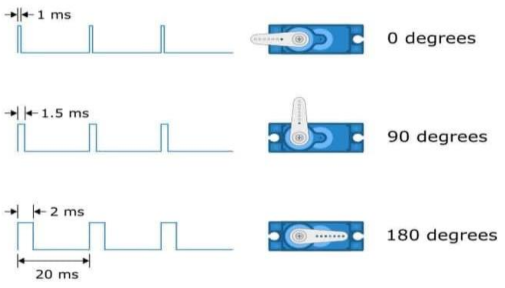

Fig (3): Servo pulses

3.2.2. Continuous servo motor pinout

The pinout for the 360-degree servo is the same as the universal, 180-degree servo motor.

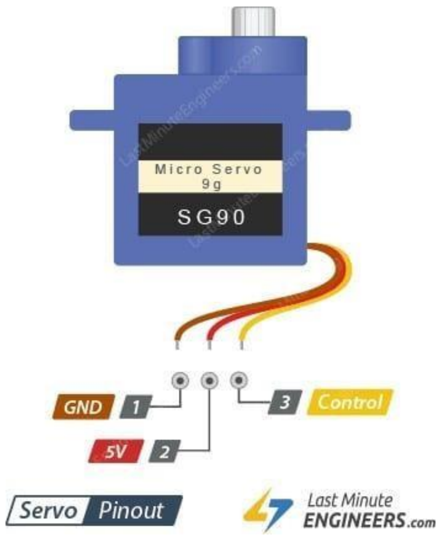

Fig (4): Servo Pinout

The brown and red pins are designated as the power pins, whereas the yellow pin acts as the signal input pin. In certain cases, such as with the Adafruit Feedback 360 Degree servo motor, there is an extra pin that delivers a voltage output between 0 and 5V, which corresponds to the shaft's angle ranging from 0 to 360 degrees. If you have this specific type of servo motor, you can ascertain the exact position of the shaft at any given time by measuring the analog Read () value from that pin.

3.3. KeyPad:

Fig (4): 4x4 keypad

4X4 keypad modules come in various sizes and shapes; however, they all share the same pin configuration. Constructing a 4X4 keypad by arranging 16 buttons in a matrix format is a straightforward process that can be done independently.

Pin Number	Description
ROWS	
1	PIN1 is taken out from 1st ROW
2	PIN2 is taken out from 2nd ROW
3	PIN3 is taken out from 3rd ROW
4	PIN4 is taken out from 4th ROW
COLUMN	

5	PIN5 is taken out from 1st COLUMN
6	PIN6 is taken out from 2nd COLUMN
7	PIN7 is taken out from 3rd COLUMN
8	PIN8 is taken out from 4th COLUMN

Table 1 : all keypad datasheet

As indicated in the table above, a 4X4 keypad consists of eight terminals. Among these, four serve as the rows of the matrix, while the remaining four function as the columns. These eight pins are derived from the 16 buttons located on the module. The 16 alphanumeric characters displayed on the surface of the module are organized in a matrix configuration.

The **internal structure of 4X4 KEYPAD MODULE** is shown below.

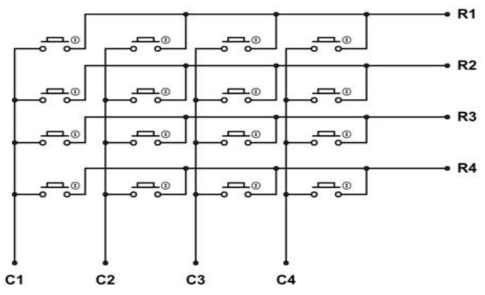

Fig (5): internal structure of 4X4 KEYPAD MODULE

3.4. Relay:

Fig (6): Relay 5 volt

A relay is a type of electro-mechanical device that operates as a switch. When the relay coil is powered by direct current (DC), it enables the opening or closing of contact switches. Typically, a single-channel 5V relay module consists of a coil and two types of contacts: normally open (NO) and normally closed (NC). This article provides an overview of the 5V relay module and its operation; however, before delving into the specifics of the relay module, it is essential to understand what a relay is and its pin configuration.

3.4.1. What is a 5V Relay?

A 5V relay functions as an automatic switch frequently utilized in automatic control circuits, enabling the management of high-current loads through a low-current signal. The input voltage for the relay signal varies between 0 and 5V.

3.4.2.5V Relay Pin Configuration

The pin configuration of the 5V relay is shown below. This relay includes 5pins where each pin and its functionality are shown below.

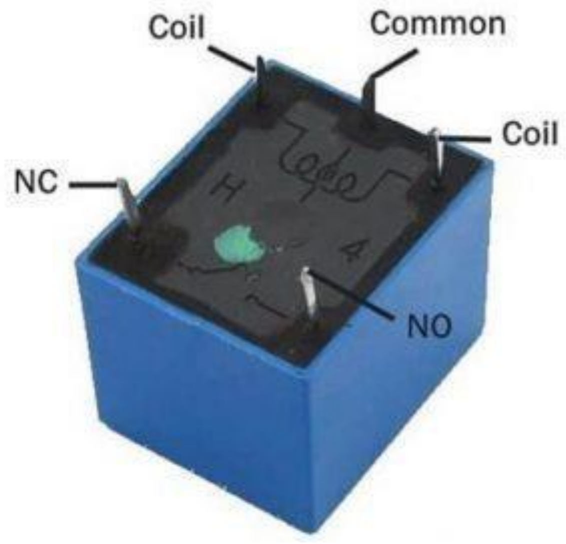

Fig (7): Relay Pin Diagram

3.4.3. How to Use/Relay Module Circuit Diagram

The circuit diagram for the single-channel relay module is presented below. This diagram illustrates the process by which the relay module is activated and deactivated via a digital signal. This signal is directed to a control pin on the relay module. Below is the internal diagram of the 5V single-channel relay module.

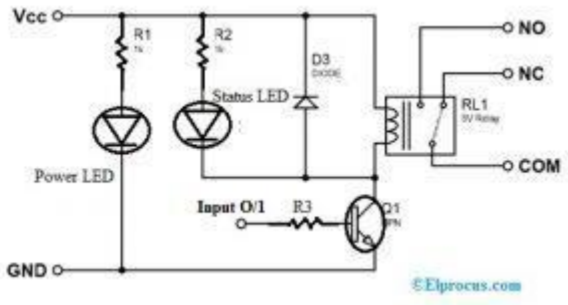

Fig (8): Single Channel Relay Module Circuit

In the circuit diagram presented above, the single-channel relay module comprises two resistors, transistors, two LEDs, and a 5V relay. Relay modules are categorized into two types based on the type of control signal utilized for relay activation. One type features an NPN transistor, while the other is equipped with a PNP transistor. When the relay module incorporates an NPN transistor, it activates the relay by applying an active high signal to the control pin. Conversely, if a PNP transistor is employed, the relay is activated by an active low signal on the control pin.

In the Proteus simulation software, when an active high signal is applied to the control pin of the relay module, the relay coil is energized, rendering the relay operational by connecting the NO pin to the COM pin. Similarly, when an active low signal is sent to the relay's control pin, the coil is deactivated using a freewheeling diode, resulting in the relay being turned off. For the PNP-based relay module, activation occurs through an active low signal, while an active high signal will lead to deactivation.

The control of a 5V single-channel relay module can be achieved by interfacing it with any microcontroller. This is accomplished by utilizing a GPIO pin, such as a digital output pin, which provides active high and low signals to the control pin. Upon activation of the relay, an audible sound can be heard emanating from the module.

3.5 Water Pump:

Fig (9): Water pump

Utilize this submersible water pump for creating fountains or watering plants. It is ideal for novice projects and remarkably user-friendly. The pump operates as a DC motor, requiring a 3V power supply and consuming 100mA. When activated, it draws water through the side of its plastic casing and expels it through the tubing outlet. It is essential to keep the pump submerged in water at all times to ensure proper operation. Reversing the polarity will not convert it into a suction device; it will solely function to pump water.

Please note that this pump is not suitable for drinking water applications. While we do not have a longevity rating for this device, our testing over a week indicates satisfactory performance; however, we do not recommend it for long-term use. It is particularly well-suited for students and artists seeking a straightforward and cost-effective solution. The motor can be controlled using PWM to adjust the flow rate, and it works effectively with either a basic power transistor for on/off control or a motor driver chip like the L293D.

3.6. Power Supply:

Fig (10): lithium battery

Lithium batteries are rechargeable batteries known for their high energy density and long lifespan compared to traditional batteries. They are used in electronic devices, electric vehicles, and renewable energy systems due to their lightweight and high efficiency. They operate using lithium-ion technology, where ions move between electrodes during charging and discharging, making them more stable and effective.

4.Block Diagram:

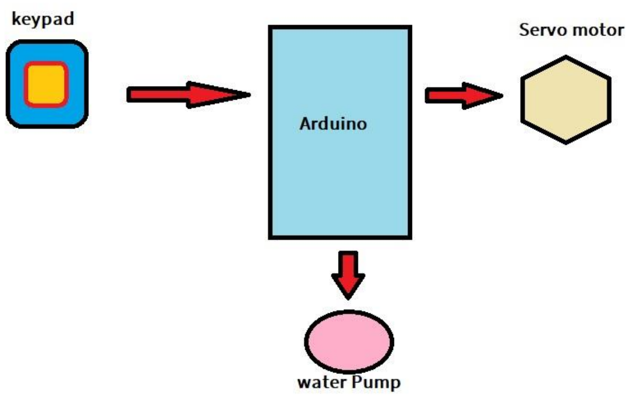

Fig (11): Block diagram

In this diagram, the operation of the circuit and the method of input and output are explained, as it begins by giving instructions to the Arduino through the keyboard by giving the command to the servo to rotate and then stopping it from the same place. After that, the pump is instructed to work to fill the bottles.

Now we will explain how each part is connected to the Arduino and programmed:

4.1. Servo motor with Arduino:

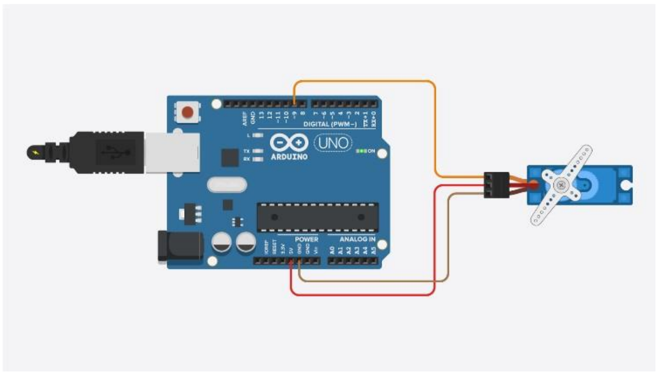

Fig (12): Servo motor with Arduino

Quackery: The servo motor is connected, as in the figure above, to the Arduino, and from the figure we can distinguish that the servo will move automatically by giving it the programming code. It is clear that there is no controller, such as a variable resistance or a switch to control it, but these examples are an explanation for us. In order to know how to deal with such motors and connect them to the Arduino microcontroller, where we can control it with precision, professionalism, and basic code.

4.2. Programming:

It is worth noting that we can program the Arduino in several languages, and one of the best and most common is C Plus, which is through the Java language. The Arduino program was designed in order to program the Arduino in the C language.

// Include the servo library:

#include "Servo.h>// Create a new servo object:

Servo myservo;// Define the servo pin:

#define servoPin 9 void setup() {

```
 // Attach the Servo variable to a pin:

 myservo.attach(servoPin);
}

void loop() {
 // Tell the servo to go to a particular angle:
 myservo.write(90); delay(1000); myservo.write(180);
 delay(1000);
 myservo.write(0); delay(1000);
 // Sweep from 0 to 180 degrees:
 for (int angle = 0; angle <= 180; angle += 1) {
 myservo.write(angle);
 delay(15);
  }
 // And back from 180 to 0 degrees:
 for (int angle = 180; angle >= 0; angle -= 1) { myservo.write(angle); delay(15); } delay(1000);
}
```

From the code above we notice that the servo motor moves at a speed of 15 milliseconds from zero to 180 degrees and then returns in the opposite direction after the same period.

4.3. Keypad with Arduino:

Keypad is used as an input device to read the key pressed by the user and to process it.4x4 keypad consists of 4 rows and 4 columns. Switches are placed between the rows and columns.A key press establishes a connection between the corresponding row and column, between which the switch is placed.

For more information about keypad and how to use it, refer the topic **4x4 Keypad** in the sensors and modules section.

Connection Diagram of 4x4 Keypad with Arduino UNO

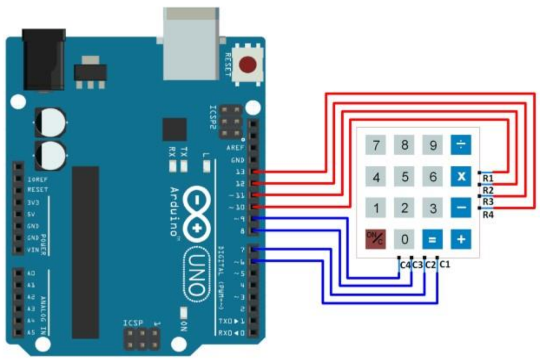

Fig (13): Interfacing 4x4 Keypad with Arduino UNO

Codding:

```
#include <Keypad.h>

        byte ROWS =  4;   const/* four rows */
        byte COLS =  4;   const/* four columns */
/*                        define the symbols on the buttons of the
keypads */  char hexaKeys [ROWS][COLS] = {
  {'0','1','2','3'},
  {'4','5','6','7'},
  {'8','9','A','B'},
  {'C','D','E','F'}
};
byte rowPins [ROWS] = {10, 11, 12, 13}; /* connect to the row pinouts of the
keypad */
byte colPins [COLS] = {6, 7, 8, 9}; /* connect to the column pinouts of the keypad
*/

/* initialize an instance of class NewKeypad */
Keypad customKeypad = Keypad(makeKeymap(hexaKeys), rowPins,
colPins, ROWS, COLS);

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

} void
  loop    (){
  char  customKey = customKeypad.getKey();

  if  (customKey){Serial.println(customKey);
```

4.4. Relay or Water pump with Arduino:

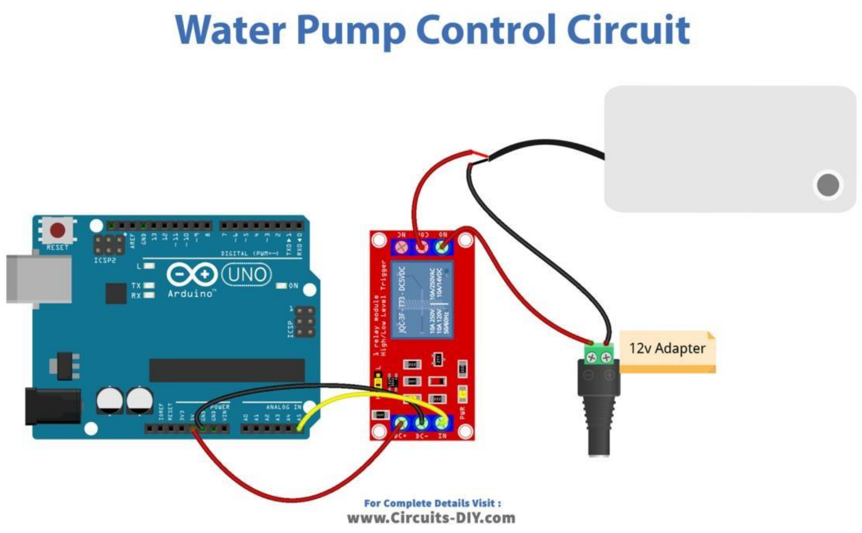

Fig (14): Water pump with Arduino

Codding:

const int RELAY_PIN = A5; // the Arduino pin, which connects to the IN pin of relay

```
// the setup function runs once when you press reset or power the board void setup() {
 // initialize digital pin A5 as an output.
 pinMode(RELAY_PIN, OUTPUT);
}

// the loop function runs over and over again forever
void loop() { digitalWrite(RELAY_PIN, HIGH); // turn on pump 5 seconds delay(5000);
digitalWrite(RELAY_PIN, LOW); // turn off pump 5 seconds delay(5000);
}
```

Now we connect all components in one circuit diagram to see full project in this shown below:

4.5. Project Circuit Diagram:

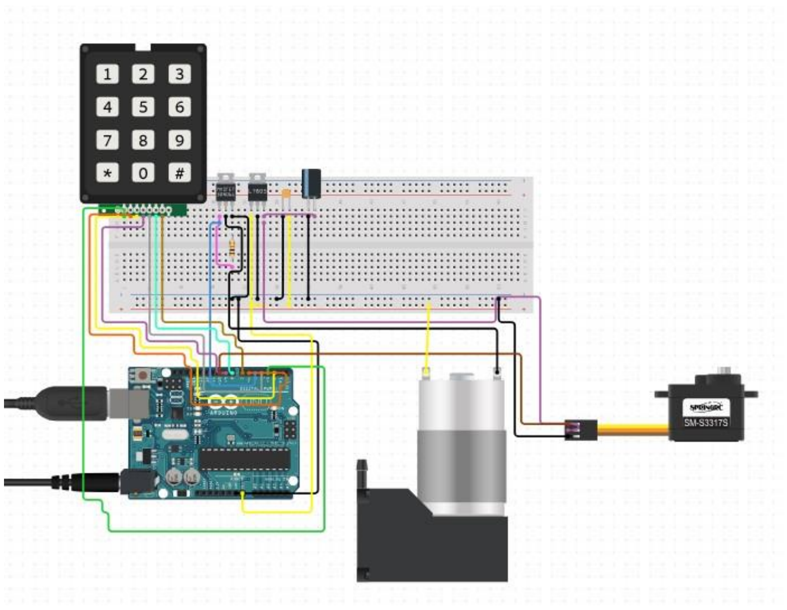

Fig (15): Project Circuit Diagram

The brown and red pins are identified as the power pins, while the yellow pin serves as the signal input pin. In specific instances, such as with the Adafruit Feedback 360 Degree servo motor, an additional pin is provided that outputs a voltage between 0 and 5V, reflecting the angle of the shaft, which varies from 0 to 360 degrees. If you possess this particular model of servo motor, you can determine the precise position of the shaft at any moment by measuring the analog Read () value from that pin.

4.6. Project Code:

```
/* @file CustomKeypad.pde
|| @version 1.0
|| @author Alexander Brevig
|| @contact alexanderbrevig@gmail.com
||
|| @description
|| | Demonstrates changing the keypad size and key values.
|| #
*/
#include <Keypad.h>
#include <Servo.h> Servo s; const byte ROWS = 4; //four rows const byte COLS = 4; //four
columns
//define the cymbols on the buttons of the keypads char hexaKeys[ROWS][COLS] = {
 {'1','2','3','A'},
 {'4','5','6','B'},
 {'7','8','9','C'},
 {'*','0','#','D'}
}; byte rowPins[ROWS] = {2,3,4,5}; //connect to the row pinouts of the keypad byte
colPins[COLS] = {6,7,8,9}; //connect to the column pinouts of the keypad

//initialize an instance of class NewKeypad
Keypad customKeypad = Keypad( makeKeymap(hexaKeys), rowPins, colPins, ROWS, COLS);
int k =13; void setup(){ Serial.begin(9600); pinMode(k, OUTPUT); s.attach(10);
}

void loop(){ char customKey = customKeypad.getKey();

 if (customKey){
 Serial.println(customKey);
 } if (customKey == '1'){ s.write(90); delay(100);
 } if (customKey == '2'){ s.write(180); } if (customKey == '3'){ s.write(270);
 }
 if (customKey == 'A'){ s.write(360);
 } if (customKey == '0'){ s.write(0);
 }
 if(customKey =='#'){ digitalWrite(k, HIGH);
 }
 if(customKey =='*'){ digitalWrite(k, LOW);
 }
}
```

5. Results and Dissociation:

As a result, a project was obtained, the least we can say is that it was successful. Through this project, the results can be read in the form of questions and answers, as follows:

1- Is it possible to find another way to control the device?

The answer: Yes, by activating the rest of the keys.

2- Can the device expand and take more than 4 bottles?

Answer: Yes, it is possible, but the type of servo motor or any other motor must be changed, but with greater torque and more power for the purpose of bearing the weights placed on it.

3- How does the device look in this position or condition in terms of bearing weight?

Answer: The device is very stable and there are no other problems 4- What are the disadvantages of this device?

Answer: There are no disadvantages, but we can say that the cost seems high because we used advanced parts for the device in manufacturing 5- Can anyone use the device?

Answer: Yes, anyone can use the device, and especially based on our instructions, we can say that children over five years old can use it as well.

6- What distinguishes the device?

Answer: It is characterized by its light weight, ease of use, and low energy consumption.

References:

1. Maker guid web site https://www.makerguides.com/servo-arduino-tutorial/
2. Circuitio..websitehttps://www.circuito.io/app?components=8653,9442,10189,103 98,11021
3. Github web site https://raw.githubusercontent.com/BasOnTech/ArduinoBeginners-EN/master/E26-4x4-matrix-keypad/E26-4x4-matrix-keypad.png
4. Arduino website https://arduinogetstarted.com/tutorials/arduino-controls-pump
5. Alprocus website https://www.elprocus.com/5v-relay-module/
6. *"Arduino UNO for beginners - Projects, Programming and Parts". makerspaces.com. 7 February 2017. Retrieved 4 February 2018.*
7. *"Arduino FAQ". 5 April 2013. Archived from the original on 27 November 2020. Retrieved 21 February 2018.*
8. Jump up to:[a b] *"What is Arduino?". learn.sparkfun.com. Retrieved 4 February 2018.*
9. Jump up to:[a b] *"Introduction to Arduino" (PDF). princeton.edu. Archived from the original (PDF) on 3 April 2018. Retrieved 4 February 2018.*
10. *"Arduino Nano". Arduino Official Store. Retrieved 2022-12-07.*
11. *"Arduino Leonardo with Headers". Archived from the original on 202105-15.*
12. *"Previous IDE Releases". Retrieved 2023-02-08.*
13. *"Arduino Older Boards". Retrieved 2023-02-08.*
14. Jump up to:[a b c d e f g h] *"Board; Uno R3; Store". Arduino.*
15. Jump up to:[a b] *Hernando Barragán (2016-01-01). "The Untold History of Arduino". arduinohistory.github.io. Retrieved 2016-03-06.*
16. *"Introducing the Arduino UNO R4! - News - SparkFun Electronics". www.sparkfun.com. Retrieved 2023-08-07.*
17. Jump up to:[a b] *"MCU; ATmega328P; Docs". Microchip. Archived from the original on March 27, 2023.*

18. *"What is Arduino UNO? A Getting Started Guide". www.rs-online.com. Retrieved 2021-08-04.*
19. *"Using Vin pin on Arduino with a shield". Electrical Engineering Stack Exchange. Retrieved 2024-01-20.*