The Remote Automatic Farm Irrigation System

Mini project, second stage semester 2

*2024-2025*

**Students names:**

Mzgeen Tajaldin

Marwan mohammad

Raad Salm

**Project supervisor:**

Dr. Omar MS Ghazal

### Acknowledgment:

The successful completion of the The Remote farm irrigation system report would not have been possible without the grace of Allah almighty firstly, and then the invaluable contributions of numerous individuals and entities. we express our deepest gratitude to Dr. Omar MS Ghazal, whose mentorship and unwavering support served as a constant source of guidance and inspiration throughout the project. His expertise in Remote farm irrigation and commitment to technological innovation provided a framework for excellence, propelling our endeavors toward fruitful outcomes.

Our heartfelt thanks extend to **Mzgeen Tajaldin and Marwan mohammad and Raad salm**, whose collaborative spirit and dedicated efforts were instrumental in overcoming challenges and maximizing our collective potential. Each member brought unique skills and perspectives to the table, fostering a synergistic environment where individual strengths merged into a cohesive force for success.

Furthermore, we acknowledge the vital role played by the UOD Department of Electric and computer Engineering. Their commitment to fostering innovation gave us the necessary resources, infrastructure, and a nurturing environment conducive to focused research and development.

Lastly, we express my deepest appreciation to my family and friends for their unwavering support and understanding during the demanding phases of this project. Their constant encouragement and unwavering belief served as a source of strength and motivation, reminding us of the significance of our work and the positive impact it holds for farmers and their families.

In conclusion, the Remote farm irrigation system report represents the culmination of a collaborative effort, with each individual and entity mentioned above playing a crucial role

in its success. I am eternally grateful for their contributions and dedicate this work to their unwavering dedication and support.

Contents

[**Acknowledgment:** 0](#_Toc21956)

[Abstract: 3](#_Toc21957)

[Chapter 1: 4](#_Toc21958)

[1. Introduction: 4](#_Toc21959)

[1.1The Remote farm irrigation System 4](#_Toc21960)

[1.2 Objectives and Goals: 5](#_Toc21961)

[1.3 Application: 5](#_Toc21962)

[1.4 Remote Farm Irrigation System (RFIS): Advantages vs. Disadvantages: 6](#_Toc21963)

[Chapter 2: 7](#_Toc21964)

[2. material and methods 7](#_Toc21965)

[2.1Components: 7](#_Toc21966)

[***Fig.1: Arduino Uno*** 7](#_Toc21967)

[***LCD 16X2 Pin Configuration***: 8](#_Toc21968)

[***Fig.2:I2C\_ LCD( 16X2)*** 8](#_Toc21969)

[**F*ig.3: The GSM 800A SIM*** 9](#_Toc21970)

[***Fig.4: The 4x4 keypad*** 10](#_Toc21971)

[***Fig.5: The buzzer*** 11](#_Toc21972)

[***Fig.6: The LED*** 11](#_Toc21973)

[***Fig.7: The Stepper Motor& Driver Motor*** 12](#_Toc21974)

[Chapter 3: 13](#_Toc21975)

[3. How does the project work? 13](#_Toc21976)

[3.1 project picture: 14](#_Toc21977)

[3.2 Electrical Circuit: 15](#_Toc21978)

[Chapter 4: 16](#_Toc21979)

[4.1Block Diagram: 16](#_Toc21980)

[4.2 Flowchart: 17](#_Toc21981)

[Chapter 5: 19](#_Toc21982)

[5. Softwere: 19](#_Toc21983)

5.1 mobile application……………………………………………………………………………………..13

5.2 windows application……………………………………………………………………………………14

5.3 microcontroller program…………………………………………………………………………

[Chapter 6: 23](#_Toc21984)

[6. Future of RASIF: 23](#_Toc21985)

[Conclusion: 23](#_Toc21986)

[Reference 24](#_Toc21987)

# **Abstract:**

The Remote Farm Irrigation System represents a transformative approach to agricultural water management. Utilizing advanced 5G-IoT technology, this system allows for precise control and monitoring of drip irrigation processes, ensuring water is delivered directly to the root zones of crops, thereby minimizing evaporation and maximizing efficiency. The integration of cloud servers and mobile applications enables users to operate the system remotely, overcoming traditional limitations of distance and range. This innovation not only reduces labor costs but also significantly improves agricultural productivity

**Chapter 1:**

**1. Introduction:**

**1.1The Toddler Diabetes Checker/Reminder System**

In the wake of escalating water scarcity and the urgent need for sustainable agricultural practices, the Remote Farm Irrigation System (RFIS) emerges as a pivotal innovation. This system is a confluence of precision agriculture, water resource management, and information technology, designed to revolutionize the way we irrigate our crops.

At its heart, the RFIS is an automated irrigation system that utilizes real-time data from various sensors deployed across the farm. These sensors monitor soil moisture levels, weather conditions, and crop water requirements, transmitting the data to a centralized control unit. Farmers can access this information through a cloud-based platform, enabling them to make informed decisions about when, where, and how much to irrigate.

The RFIS is not just about conserving water; it’s about optimizing crop yield and ensuring food security in a world where the agricultural landscape is being reshaped by climate change. By applying the precise amount of water needed for crops to thrive, the system reduces runoff, decreases soil erosion, and prevents nutrient leaching, thereby maintaining the ecological balance of the farmland.

Moreover, the RFIS is equipped with advanced features such as remote-control valves, automated scheduling, and predictive analytics. These features allow for a hands-off approach to irrigation management, freeing up time for farmers to focus on other critical aspects of their operations. The system’s adaptability to various crop types and farming scales makes it a versatile tool for both smallholder farmers and large-scale agricultural enterprises.

As we look to the future, the RFIS stands as a testament to human ingenuity and our collective effort to create a sustainable and prosperous world for generations to come. It is not just a system; it is a movement towards smarter, more responsible farming—a movement that begins with a single drop of water used wisely.

1.2 **Objectives and Goals:**

**Objectives:**

* **Water Conservation**: To minimize water wastage by delivering water precisely where and when it is needed, contributing to the conservation of this vital resource.
* **Enhanced Crop Yield**: To increase agricultural productivity by ensuring that crops receive the optimal amount of water at the right times.
* **Operational Efficiency**: To reduce labor costs and time spent on irrigation through automation and remote monitoring capabilities.
* **Sustainability**: To promote sustainable farming practices by optimizing water usage and reducing the environmental impact of irrigation.

**Goals:**

* **Remote Monitoring and Control**: To enable farmers to monitor and control their irrigation systems from any location, thereby improving management and responsiveness.
* **Data-Driven Decisions**: To provide actionable insights through data collected from sensors, allowing for informed decision-making regarding irrigation practices.
* **Resource Optimization**: To ensure efficient use of water and energy resources, leading to cost savings and environmental benefits.
* **Adaptability and Scalability**: To design the system to be adaptable to various farm sizes and types of crops, making it a versatile tool for a wide range of agricultural needs.

**1.3 Application:**



Applications for a Remote Farm Irrigation System can be quite diverse, addressing various aspects of farm management and crop cultivation. Here are some key applications:

### 1. ****Water Management****

* **Efficient Water Use**: Utilizes sensors to monitor soil moisture and weather conditions to apply the right amount of water.
* **Leak Detection**: Identifies leaks in the irrigation system to prevent water loss.
* **Irrigation Scheduling**: Automates watering schedules based on crop needs and environmental data.

### 2. ****Crop Monitoring****

* **Growth Tracking**: Monitors crop growth stages to optimize irrigation and fertilization.
* **Health Assessment**: Detects signs of plant stress or disease early through remote sensing..
* **Yield Prediction**: Uses data analytics to forecast crop yields and plan for harvest.

### 3. ****Resource Optimization****

* **Energy Savings**: Reduces energy consumption by optimizing pump usage and irrigation timing.
* **Cost Reduction**: Lowers operational costs by minimizing manual labor and resource wastage.
* **Input Management**: Manages the application of water, fertilizers, and pesticides more effectively.

### 4. ****Data Analysis and Decision Support****

* **Real-Time Data**: Provides up-to-date information on farm conditions for informed decision-making.
* **Historical Data Analysis**: Analyzes past data to improve future irrigation strategies.
* **Predictive Analytics**: Employs machine learning algorithms for better resource planning.

### 5. ****Environmental Conservation****

* **Sustainable Practices**: Supports eco-friendly farming by reducing the carbon footprint of irrigation.
* **Soil Preservation**: Helps maintain soil structure and fertility by preventing over-irrigation.
* **Biodiversity Protection**: Minimizes the impact on local ecosystems through precise irrigation.

These applications demonstrate how a Remote Farm Irrigation System can significantly contribute to modern, efficient, and sustainable agriculture.

**1.4 Remote Farm Irrigation System (RFIS): Advantages vs. Disadvantages:**

|  |  |
| --- | --- |
| **Advantages** | **Disadvantages** |
| Efficient water usage | Initial cost |
| Reduced water waste | Dependence on technology |
| Improved crop yield | Maintenance requirements |
| Precision in irrigation scheduling | Connectivity issues |
| Remote monitoring and control | Vulnerability to cyber threats |
| Flexibility in irrigation management | Learning curve for users |
| Labor savings | Potential data breaches |

**Chapter 2:**

**2. material and methods**

This chapter showcases the essential components that power our project. Each element, carefully wired and programmed to interact with the Arduino, plays a specific role in fulfilling our overall vision. By working together, they seamlessly execute the planned functionality, bringing our concept to life.

**2.1Components:**

**1. The Arduino Uno**

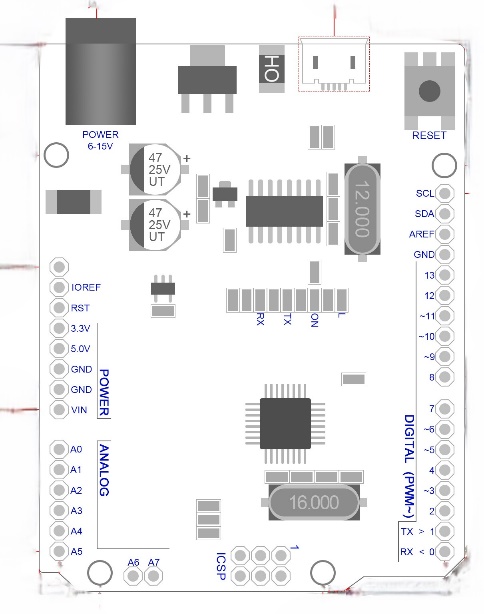
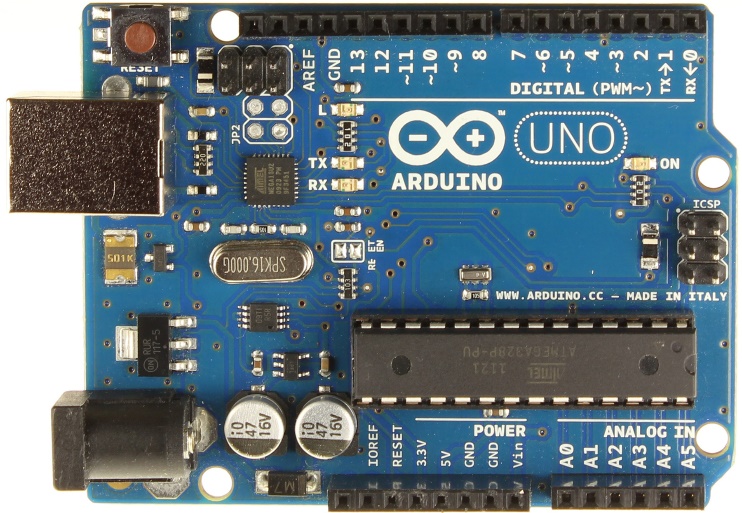
An enduring fixture in the realm of microcontrollers, the Arduino Uno shines with its robust capabilities and accessible design. Driven by the ATmega328 microcontroller, this board strikes a harmonious balance between simplicity and functionality, catering to both novices and seasoned developers alike.

With 14 digital input/output (I/O) pins and 6 analog input pins, the Uno offers a versatile platform for a myriad of projects, albeit with fewer I/O pins compared to its Mega 2560 counterpart. Despite this, the Uno remains a stalwart choice for electronics enthusiasts, providing ample connectivity for sensors, actuators, and peripheral devices. Its compact form factor and user-friendly interface make it an ideal tool for educational endeavors and rapid prototyping ventures.

Despite its smaller footprint, the Uno doesn't compromise on performance, boasting an 8-bit architecture and a clock speed of 16 MHz to efficiently handle various algorithms and real-time tasks.

While newcomers may encounter a slight learning curve, the Uno's extensive documentation and supportive community ensure ample resources for mastering its capabilities.

For a deeper dive into its features and specifications, consulting official Arduino documentation is highly recommended.



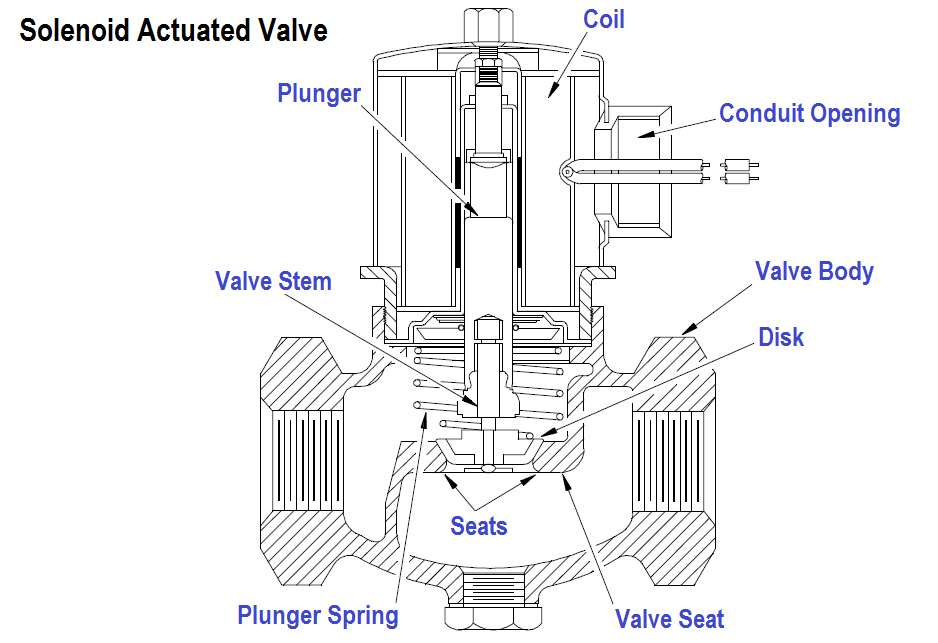
***Fig.1: Arduino Uno***

**2. solenoid valves**

A solenoid valve is an electromechanically operated valve.

Solenoid valves differ in the characteristics of the electric current they use, the strength of the magnetic field they generate, the mechanism they use to regulate the fluid, and the type and characteristics of fluid they control. The mechanism varies from linear action, plunger-type actuators to pivoted-armature actuators and rocker actuators. The valve can use a two-port design to regulate a flow or use a three or more port design to switch flows between ports. Multiple solenoid valves can be placed together on a manifold.

Solenoid valves are the most frequently used control elements in fluidics. Their tasks are to shut off, release, dose, distribute or mix fluids. They are found in many application areas. Solenoids offer fast and safe switching, high-reliability, long service life, good medium compatibility of the materials used, low control power and compact design.



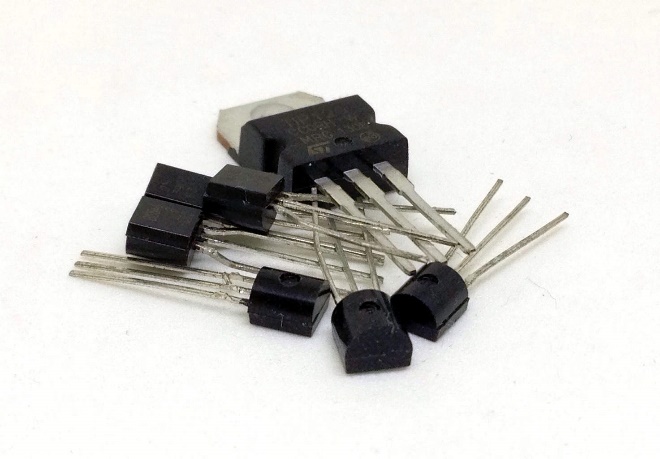


***Fig.2: solenoid valves***

**3. transistors**

A transistor is a type of semiconductor device that can be used to conduct and insulate electric current or voltage. A transistor basically acts as a switch and an amplifier. In simple words, we can say that a transistor is a miniature device that is used to control or regulate the flow of electronic signals.

Transistors are one of the key components in most of the electronic devices that are present today look at fig 3. Developed in the year 1947 by three American physicists, John Bardeen, Walter Brattain and William Shockley, the transistor is considered one of the most important inventions in the history of science.



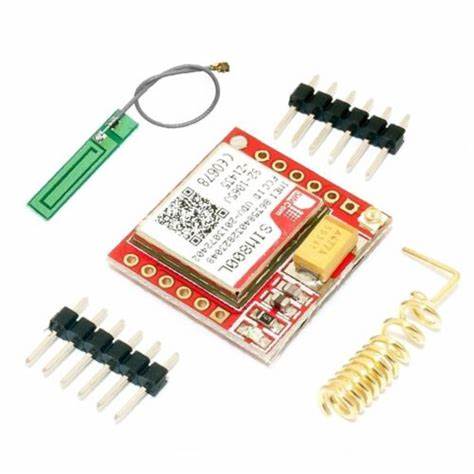
***Fig.3: Transistor***

**4. Sim808L modules**

The SIM808L is a GSM/GPRS module designed for use in embedded systems. It provides voice, SMS, and data connectivity for applications requiring remote communication. The module is small in size and low in power consumption, making it suitable for use in portable and battery-powered devices.

The SIM808L operates on quad-band GSM/GPRS networks, meaning it can be used worldwide. It supports GPRS Class 10 data transfer up to 85.6 kbps for upload and download. It also has an integrated TCP/IP stack, allowing it to communicate with the Internet. The module includes a serial interface for communication with a microcontroller, and it supports both AT commands and GPRS commands.

The SIM808L can be used for various applications, including vehicle tracking, remote monitoring and control, and home automation. It can also be used in industrial applications, such as remote sensing and control, and in medical devices, such as remote patient monitoring. Overall, the SIM800L is a reliable and versatile GSM module that provides connectivity for a wide range of applications



F***ig.4: Sim 808L***

**5. SD card reader module**

Secure Digital, often abbreviated as SD, is a non-volatile memory card format that’s used extensively in portable devices such as digital cameras, smartphones, and tablet computers. The SD card standard is maintained by the SD Association. An SD Card Module, on the other hand, is a device that allows microcontrollers, like Arduino or Raspberry Pi, to read and write data to/from SD cards.

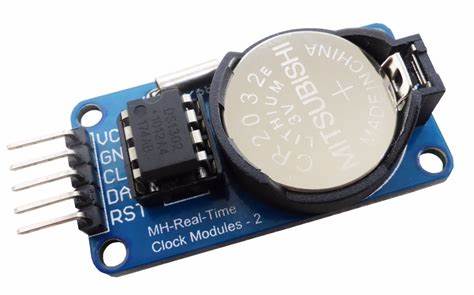


***Fig.5: SD card reader module***

**6. RTC module**

A real-time clock (RTC) is an integrated circuit clock module usually found in modern computers, servers, or embedded systems. Also, the RTC module has one purpose; timekeeping.

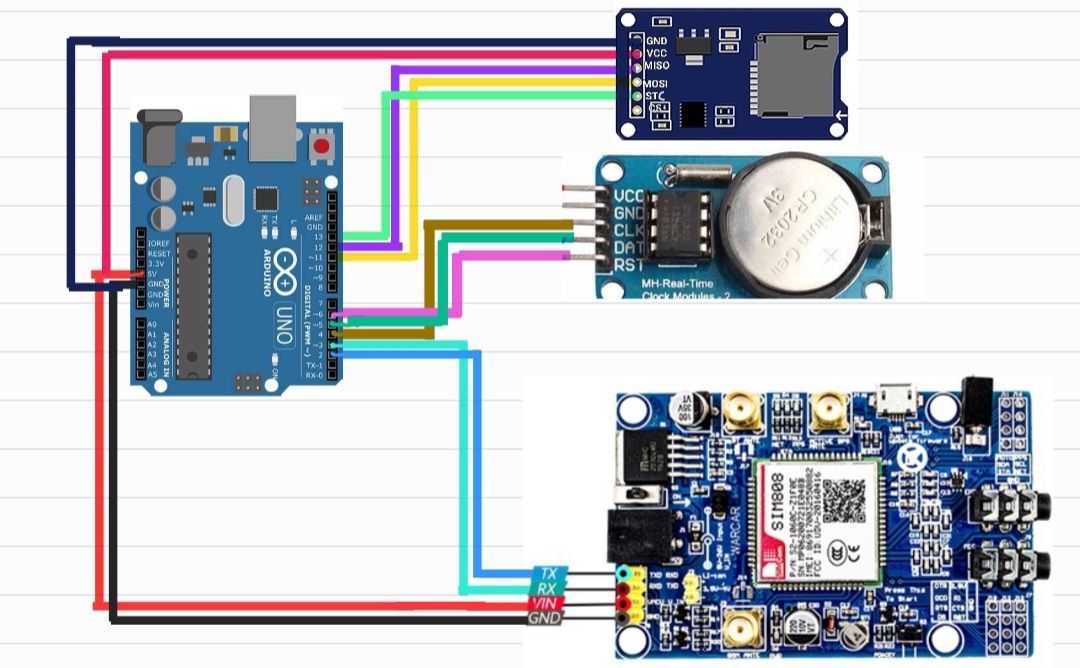
This device can count seconds, minutes, hours, and years. Plus, it’s capable of handling various time-keeping applications accurately.

****

***Fig.6: RTC module***

**2.2 Circuit diagram:**

We divided the diagram into two diagrams for clarity, so we have **A** and **B**

****

**fig 2.A**

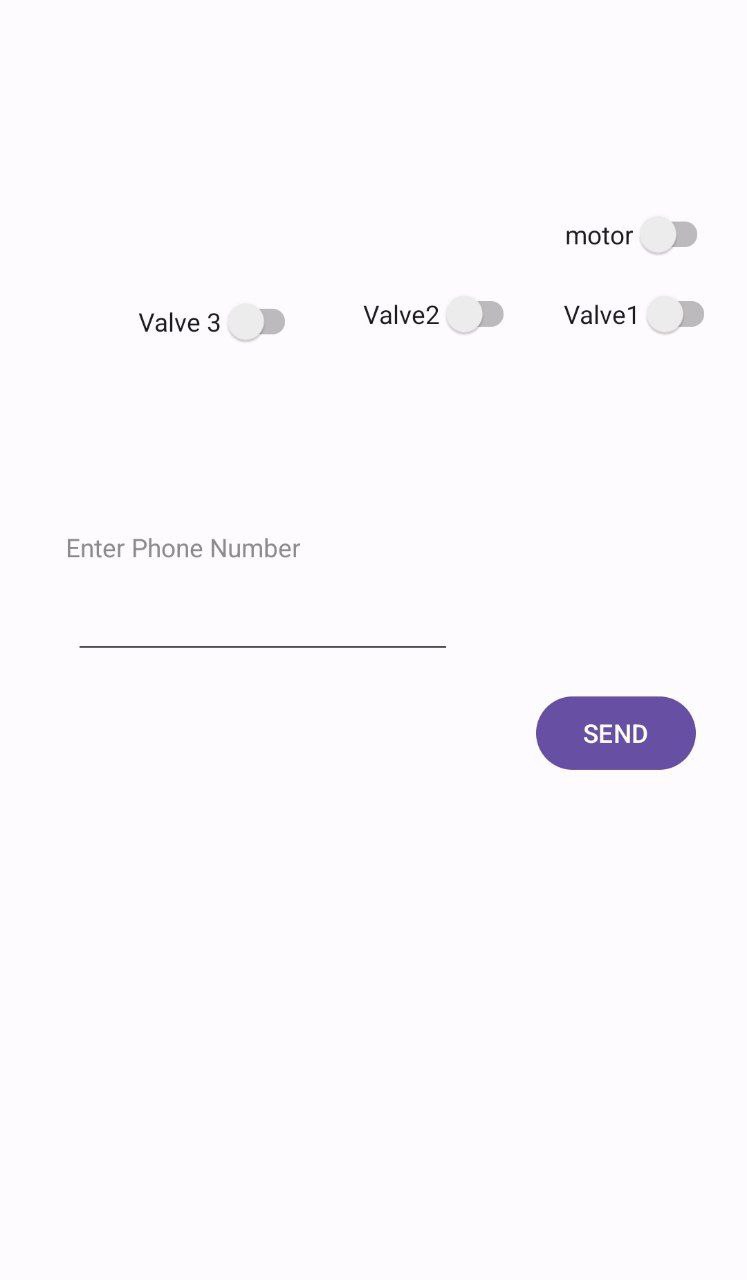
fig 2.A shows the main digital components of the circuit, and here is the pin connections

|  |  |
| --- | --- |
| **SD reader** | **Arduino Uno** |
| MISO | Digital 12 |
| MOSI | Digital 11 |
| STC | Digital 13 |
| **Real Time Clock** |  |
| CLK | Digital 4 |
| DAT | Digital 6 |
| RTS | Digital 5 |
| **GSM SIM 808** |  |
| TX | Digital 2 |
| RX | Digital 3 |
| **COMMON** |  |
| VCC | 5V |
| GND | GND |

**Chapter 5: software**

**5.1. Mobile Application**

The application consists of single functionality i.e. sending SMS massages, the application interface is very simple, it consist of a text box where the user would enter the number of the SIM card , and by turning checkbox on and off, then by clicking the SEND button , am massage will be sent by the Timecard to that number containing information about witch valve to open and witch to close



*Fig 5.1 : application interface*

The coding was done in java programming language and it supports android devices

String n = "";//an empty string to save the information before sending

//defines the interface elements in the code  
Switch motorS = (Switch) findViewById(R.id.*motor*);  
Switch v1 = (Switch) findViewById(R.id.*V1*);  
Switch v2 = (Switch) findViewById(R.id.*V2*);  
Switch v3 = (Switch) findViewById(R.id.*V3*);  
EditText num = (EditText) findViewById(R.id.*Number*);

//saves on valves by 1 and off valves by 0

n = n + ((motorS.isChecked()) ? "1":"0");  
n = n + ((v1.isChecked()) ? "1":"0");  
n = n + ((v2.isChecked()) ? "1":"0");  
n = n + ((v3.isChecked()) ? "1":"0");

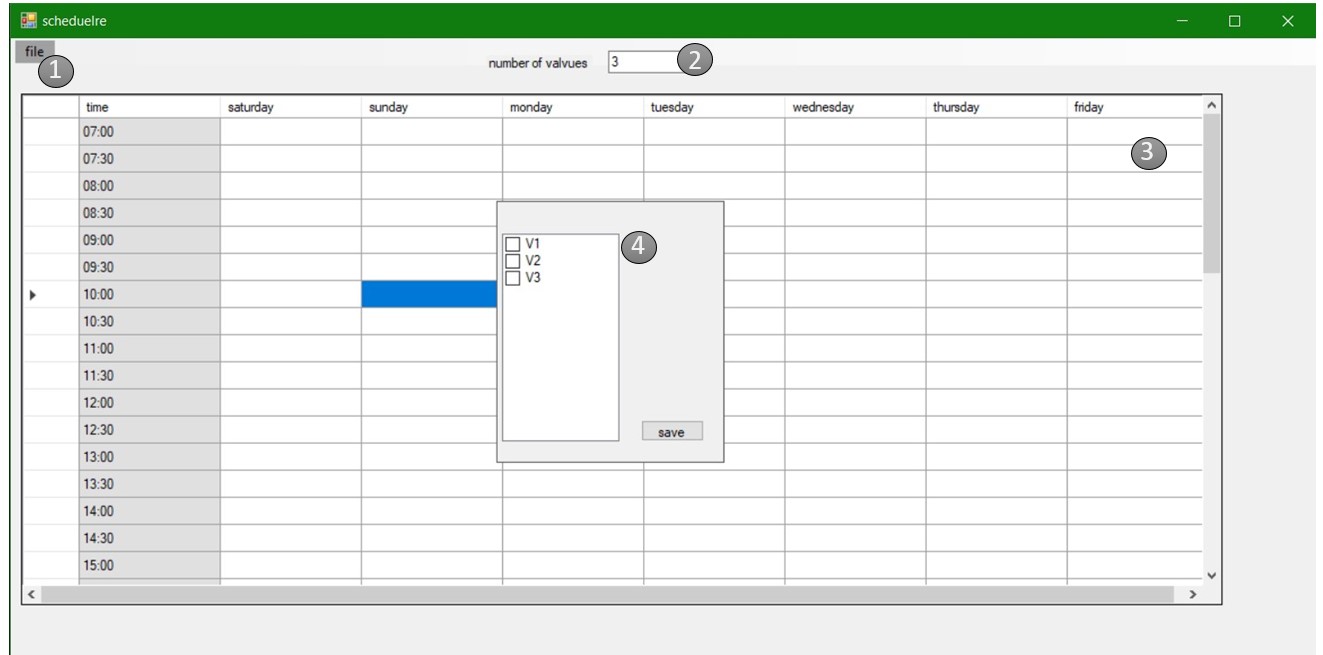
SmsManager man = SmsManager.*getDefault*();//initialize the SMS API  
String number = (String) num.getText().toString();//gets the phone number from text box  
man.sendTextMessage(number, null, n, null, null);

The last line is the most important, it is a call for the functoin sendTextMessage it has 5 parameters , only 2 of them being important for this application, the first which is the number for witch the text is to be sent to, and the third witch is the text , in our case we get the number from the text box and the text is our valve ON and OFF states.

**5.2. Windows forms application**

As discussed earlier RASIF is capable of automatic system for controlling the valves, this is done by following a premade table, this application is built in c++ specifically for constructing the table,although not all, but the fast majority of the functionalities that this application has are taught in c++/data structure lecture in ECE, and since the application very large (over 520 lines as for the last update)so it will be discussed in parts and with neglecting details unlike step by step as the mobile application:

**5.2A: User Interface**

The interface is consist of 4 main parts 

1- the menu bar: used for saving the table .

2- the valves text box, the user should enter the number of valves.

3-the table viewer, this allows the user to select time squares on the table to select operating times.

4-data pop-up , this is a panale that is showed on double clicking a time square, the user specifies witch valve should be opened.

**5.2B: Data Structure:**

time blocks are the main data type in this application, it is a data structure with following members

value struct timeBlock

{

bool\* x;

int startH, startM;

int finishH, finishM;

int day;

};

The size of the single time block is (21byte +number of valves), so for a table of 200 time blocks (such a quantity will scarcely be necessary) the size of the table is about (5Kbyte), befor saving the table is in the memory , inside a list of 200 time blocks, this small size makes it possible to run this program on a lot devices, and this structure is saved inside a .dat file with the following format

dd sh sm fh fm v1 v2 v3 v4……..

**5.2C: primary:**

In this step all the elements in the UI are declared, and initialized, this section of the code is done automatically by visual studio and it is basically as following

private: datatype VariableName;//decleration

InitializeComponent()//initialization

{

this->VariableName = (gcnew datatype);

//VariableName (setting member Variables)

VariableName->Width = 12;

VariableName->Text = “example”;

{

And so on for all the remaining mebers

**5.2D: Functions :**

**In summary:**

* scheduelre\_Load(); this function is used at the start of the program for loading time values to the table
* dataGridView1\_CellDoubleClick(); this function is called when the user double clicks on a cell used for selecting a cell in the table and showing the panale of valves
* dataGridView1\_CellClick(); this function is called when a user clicks one time on a cell used for hiding the valves panale if the user clicks away
* button1\_Click(); this function is called when the user clicks the save button on the panel used for saving a time block in the list of time blocks
* mySwap(timeBlock %block1, timeBlock %block2); this function is called inside an sorteData funcionused for exchanging two timeblcoks(used in sorting)
* sortData(); this function is used for sorting the blocks in the block list into a ascending
* saveToolStripMenuItem\_Click(); this function is used for saving all the file

**in more details**

void scheduelre\_Load()

{  
for (float i = 0; i < 24; i += 0.5)

{

dataGridView1->Rows->Add(DateTime(DateTime::Now.Year, DateTime::Now.Month, DateTime::Now.Day, (((int)i + 7) % 24), fmod(i, 1) \* 60, 0).ToString("HH:mm"));

}

}

Void dataGridView1\_CellDoubleClick(){

checkedListBox1->Items->Clear();

if (Nvalves->Text != "")

{

int n = Convert::ToInt16(Nvalves->Text);

for (int i = 0; i < n; i++)

{

checkedListBox1->Items->Add("V" + (i + 1).ToString());

}

}//pop up panel

panel1->Location = System::Drawing::Point(dataGridView1->SelectedCells[0]-> ColumnIndex \* dataGridView1->Columns[0]->Width + 180 - dataGridView1 -> HorizontalScrollingOffset, (dataGridView1->SelectedCells[0]->RowIndex \* dataGridView1-> RowTemplate->Height) - dataGridView1->VerticalScrollingOffset); //sets the position to cell

panel1->Visible = 1; }

void dataGridView1\_CellClick() //hides the valves panale on user clicks off

{

panel1->Visible = 0;

}

void mySwap(timeBlock %b1, timeBlock %b2) //used for exchanging in soring the timeBlocks

{

timeBlock temp = b1;

b1 = b2;

b2 = temp;

}

void sortData(int rows)//bubble sorting algorithm

{

for (int i = 0; i <rows;i++)//

{

for (int j = 0; j <rows-i-1; j++)

{

if(allBlocks[j+1].x != NULL)

{

if (allBlocks[j].day > allBlocks[j + 1].day)

mySwap(allBlocks[j], allBlocks[j + 1]);

else if (allBlocks[j].day == allBlocks[j + 1].day)

{

if (allBlocks[j].startH > allBlocks[j + 1].startH)

mySwap(allBlocks[j], allBlocks[j + 1]);

else if (allBlocks[j].startH > allBlocks[j + 1].startH)

if (allBlocks[j].startH > allBlocks[j + 1].startH)

mySwap(allBlocks[j], allBlocks[j + 1]);

}

}

}

}

}

**Next function will not be the full code due the length of it but rather it is a simplified version**

**Void** button1\_Click()  
{

timeBlock t;

t.x = new bool[Convert::ToInt32(Nvalves->Text)];

for (int i = 0; i < Convert::ToInt32(Nvalves->Text); i++)

t.x[i] = 0;

int row = dataGridView1->SelectedCells[0]->RowIndex;

//gets starting hour from the row of the corresponding cell selected

t.startH = ((safe\_cast<int>(dataGridView1->Rows[row]->Cells[0]->Value->ToString()[0] \* 10) + (safe\_cast<int>(dataGridView1->Rows[row]->Cells[0]->Value->ToString()[1]) )));

//same for min

t.startM = Convert::ToInt32(((dataGridView1->Rows[row]->Cells[0]->Value->ToString()[3].ToString()))) \* 10;

//day

t.day = dataGridView1->SelectedCells[0]->ColumnIndex;

//default

t.finishM = (t.startM + 30) % 60;

if (t.finishM == 0)

t.finishH = t.startH + 1;

else

t.finishH = t.startH;

To be continued below

int j = 0;

int valves = Convert::ToInt32(Nvalves->Text);

//for valves

dataGridView1->SelectedCells[0]->Style->BackColor = Color::Red;

for (int i = 0; i < valves; i++)

{

if (checkedListBox1->CheckedItems->Count == 0)

break;

if (checkedListBox1->Items[i]->ToString() == checkedListBox1->CheckedItems[j]->ToString()) {

t.x[i] = 1;

j++;

//to avoid trying to save more after selected finish but the original not finished

if (j >= checkedListBox1->CheckedItems->Count)

break;

}

}dataGridView1->ClearSelection();

panel1->Hide();

allBlocks[top] = t;

top++;

}//end of function click\_button

The following method is the hole purpose of this application it saves all information to a .dat file

void saveToolStripMenuItem\_Click()

{

statusLab->Text = "ok";

sortData(top);//pronlem null reftancve

std::fstream file("tabel1.dat”, std::ios::app);

for (int i = 0; i < top; i++)

{

file << allBlocks[i].day;

file << " ";

if (allBlocks[i].startH <= 9)

file << "0";

file << allBlocks[i].startH;

file << " ";

if (allBlocks[i].startM < 9)

file << "0";

file << allBlocks[i].startM;

file << " ";

if (allBlocks[i].finishH <= 9)

file << "0";

file << allBlocks[i].finishH;

file << " ";

if (allBlocks[i].finishM <= 9)

file << "0";

file << allBlocks[i].finishM;

file << " ";

for (int j = 0; j < Convert::ToInt32(Nvalves->Text); j++)

{

file << allBlocks[i].x[j];

file << " ";

}

file << "\n";

status Lab->Text = "saved";

}

**Chapter 3:**

**How does the project work?**