Intelligent Irrigation System

(2022-2023)

Course Code: Real Time Embedded Programming

Course Name: ENG 5220

Group Number: Group

Lecturer: Bernd Porr & Nick Bailey

Team members: Wei Wei, Xiaoyu Qian, Jinhai Hu, Meiwen Li

University of Glasgow

Content

University of Glasgow

1. Introduction:

As plant enthusiasts, it can be difficult to take care of plants when traveling or away from home for extended periods of time. This often leads to plants not receiving the necessary care they require, resulting in poor growth or even death. To address this issue, a smart and automated plant watering system can be implemented to ensure that plants receive the necessary care even when owners are away from home. This system can be controlled remotely through a mobile application or a web interface.

In addition to the challenge of the long distance between the host and the plant, extreme weather conditions can also pose a threat to plants. High winds, heavy rain, snow, and hail can damage plants and even uproot them. Therefore, it is essential to provide additional protection to plants during extreme weather conditions to ensure their survival. By integrating an automated vehicle with the smart plant watering system, plant enthusiasts can move the plants to a more sheltered area and water them as needed, ensuring that they remain healthy and protected during harsh weather conditions.

Having addressed the remote-control issue for plants and plant enthusiasts, as well as the protection of plants in extreme weather conditions, we also hope to achieve water resource recycling and reuse, making the smart irrigation system environmentally conscious. The smart and automated plant watering system also includes a feature for water resource conservation and reuse. The system collects and retains excess water in a reservoir, which can be pumped back to the plants when needed. This ensures that water is used efficiently and conserves this valuable resource.

Intelligent irrigation systems are designed to address the challenges that plant enthusiasts face when caring for their plants while away from home. The system provides precise watering, protects plants from small animals, allows for timely watering through remote control, and conserves water resources. By implementing this system, plant enthusiasts can ensure their plants remain healthy and protected, even when they are unable to tend to them personally or away from home.

2. System Design:

Our designed system aims to address several problems, including the remote control of the irrigation system, providing a safe environment for plants during extreme weather conditions, and implementing water resource recycling and conservation. By utilizing the mechanical structure of the flowerpot and the smart car, we can implement a water resource recycling system. The system collects and stores excess water in a water tank, which can be used for irrigation when plants require watering.

Any excess water is returned to the tank for future use. The plant's status can be monitored in real-time using a web interface, allowing the smart irrigation system to be activated when necessary. In the event of external interference such as extreme weather (such as hail, heavy rain, drought, and flood) or invasion by small animals (attempting to eat plants), the smart car's dynamic obstacle avoidance and path planning functions can be activated to transport the plants to a safe location for protection. The following is an introduction to the hardware components and software of the intelligent irrigation system.

1. Hardware Design: The hardware design includes the irrigation system and the smart car; the hardware facilities of the irrigation system include soil moisture sensors, analog-to-digital converters, water pumps, external power supplies, and relays, while the hardware facilities of the smart car include sensors (ultrasonic, light-sensitive, and camera sensors), chips (PCA9685 and PCF8591), and power execution structures (servo motors and motor drivers) and so on.

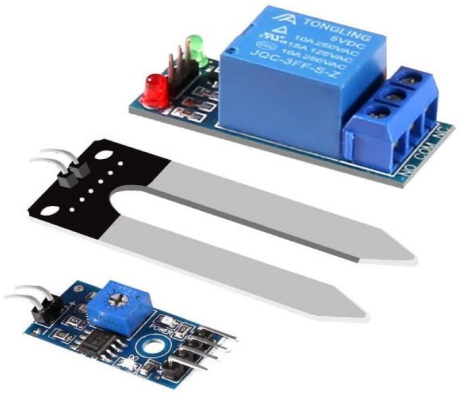

Fig 2. Soil moisture sensor

2. Software Design: Raspberry Pi, C++

These resources can form two major functional parts in our design, the smart car and the smart irrigation system, and each has its own unique function and use. The smart car provides a safe environment for plants by incorporating dynamic obstacle avoidance and path planning and is capable of transporting plants to a secure location during external disturbances. On the other hand, the smart irrigation system mainly addresses the issue of watering plants by enabling remote control of the system while simultaneously implementing water resource conservation and recycling.

Moving forward, we will discuss and analyze the software and hardware connections of the smart irrigation system and smart car separately. For the smart irrigation system, the hardware components include soil moisture sensors, analog-to-digital converters, water pumps, external power supplies, and relays. The analog-to-digital converters and relay modules can be connected to the Rapberry Pi's GPIO port to read and control signals. In terms of software, we can write C++ scripts to control the analog-to-digital converters and relay modules, enabling remote control of the irrigation system.

For the smart car system, the hardware components include ultrasonic sensors, light-sensitive sensors, camera sensors, chips (PCA9685 and PCF8591), and power execution structures (servo motors and motor drivers). The chip and power execution structure modules can be connected to the Raspberry Pi's GPIO port for control and management of the power execution structure. Similar to the irrigation system, we can write C++ scripts to control the chip and power execution structure modules, allowing for control and management of the smart car.

University of Glasgow

3. Irrigation System

3.1 Hardware components

The hardware design includes the irrigation system and the smart car; the hardware facilities of the irrigation system include soil moisture sensors, analog-to-digital converters, water pumps, external power supplies, and relays.

3.2 Hardware assembly and implementation

Soil moisture sensor is a type of sensor used to measure soil moisture, typically composed of two electrodes and a circuit board. When the electrodes are inserted into the soil, the moisture in the soil will create a resistance between the two electrodes, and the circuit board measures this resistance to calculate the soil moisture. However, since the soil moisture sensor outputs an analog signal, it needs to be converted into a digital signal using an analog-to-digital converter (ADC) so that the Raspberry Pi can read and interpret the signal.

An ADC is an electronic device that converts an analog signal into a digital signal and typically communicates with the Raspberry Pi through SPI or I2C interfaces. Among them, the MCP3008 is a commonly used 8-channel 12-bit ADC chip with high accuracy and stability, which can convert the analog signal of the soil moisture sensor into a digital signal. By using appropriate software libraries and drivers, the configuration and data reading of the MCP3008 can be implemented to obtain the specific value of soil moisture.

However, using MH-sensor series devices to detect soil moisture and outputting digital signals directly can make it easier for the Raspberry Pi to process and interpret data, and achieve the setting of dry and wet modes. The MH-sensor series devices can measure soil moisture and output digital signals directly, without the need for an ADC to convert analog signals into digital signals. Based on the output digital signal, we can write appropriate software programs on the Raspberry Pi to implement dry and wet modes settings.

For example, when the soil moisture is below a certain threshold, the Raspberry Pi can automatically trigger the irrigation system to water the plants,

and when the soil moisture is above another threshold, the Raspberry Pi can automatically stop the irrigation system to avoid overwatering. This way, automatic irrigation control of plants can be achieved while improving the efficiency of water resource utilization.

As for the water pump, since the output signal capability of the Raspberry Pi is limited, it cannot directly control high-power devices. Therefore, an external power supply and relay combination is required to control the water pump. A relay is an electronic device that can convert a low-level signal into a high-level signal to control the on-off of high-power devices.

When the water pump needs to be started, the Raspberry Pi outputs a high-level signal to the control input pin of the relay, which closes the relay circuit and powers the water pump to start working. When the water pump is not needed, the Raspberry Pi outputs a low-level signal to the control input pin of the relay, which opens the relay circuit and cuts off the power supply of the water pump. This way, remote control of the water pump can be achieved through the GPIO pins of the Raspberry Pi.

It should be noted that when using electrical equipment and circuits, safety regulations and standards must be followed to avoid possible fire, electric shock, or other safety hazards. For example, appropriate cables and connectors should be used to connect the sensors and converters, and the circuit should be appropriately protected and isolated. When using a relay, attention should be paid to its rated power and operating voltage to avoid exceeding its allowable range. In addition, appropriate power sources and overload protection devices should be used to ensure circuit stability and safety.

3.3 Software design and implementation

Based on the flowchart presented in Figure 1, we have developed a system that utilizes a soil moisture sensor to accurately detect the moisture level of a plant. This information is then used to automatically control a water pump, ensuring that the plant receives an appropriate amount of water. If the moisture level is too low, the pump is activated to deliver water to the plant, while if the level is too high, the pump is deactivated to prevent over-watering.

The program behind this system employs the wiring Pi library and GPIO control to efficiently read values from the soil moisture sensor and manage the

on/off state of the water pump. Initially, the program sets up the wiring Pi library and designates the soil moisture sensor and water pump pins as input and output modes, respectively.

Following this setup, the program enters a continuous loop, wherein each iteration involves reading the digital input value from the soil moisture sensor. Based on this value, the program determines whether the moisture level is too low or too high and adjusts the water pump's activity accordingly. This intelligent system ensures optimal watering for the plant, thus promoting healthy growth and reducing the risk of damage due to improper watering.

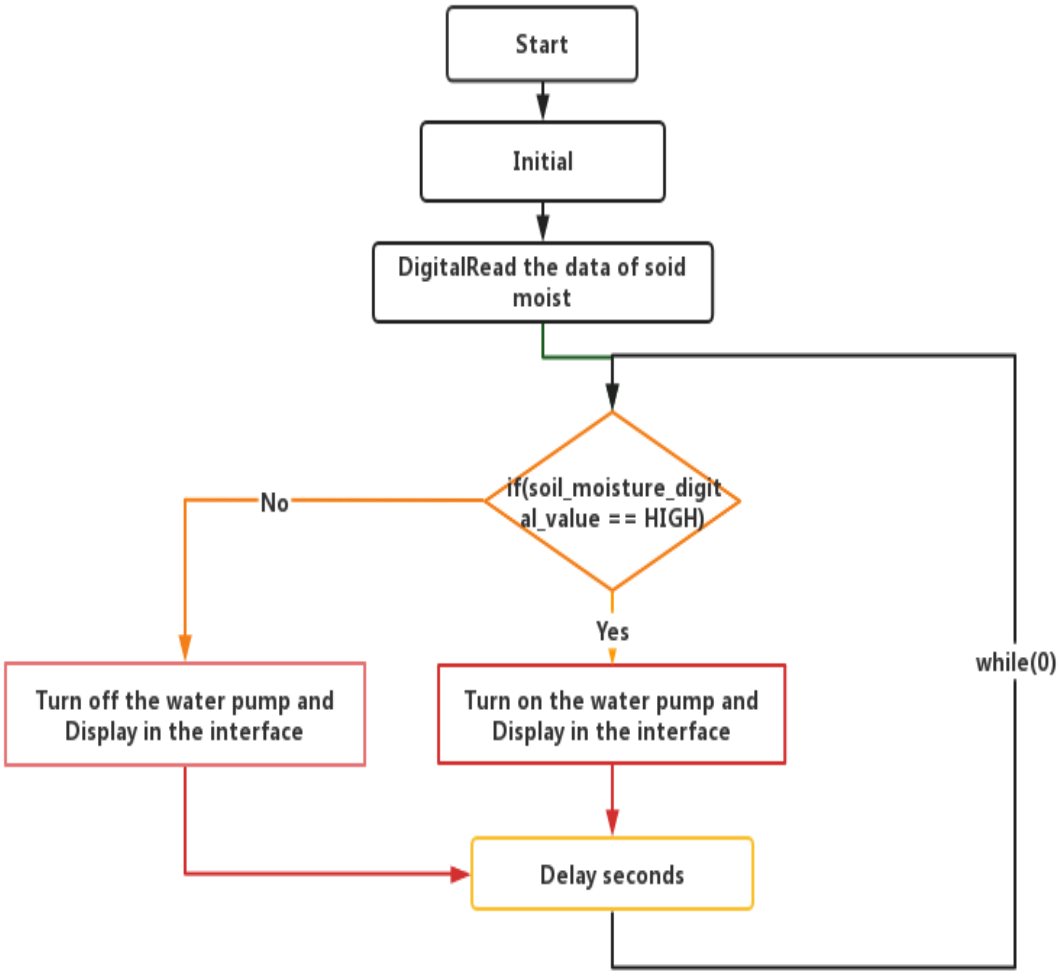

Fig 3.3. Flowchat of Irrigation system

After analyzing the software structure and flowchart, we need to perform detailed programming operations for each part. The following is a detailed explanation of the program:

The program uses the GPIO pins of the Raspberry Pi to control the water pump switch by detecting soil moisture value, thus achieving automatic watering control of plants.

At the beginning of the program, three library files are imported: iostream, wiringPi.h, and thread. iostream library is the C++ standard input and output stream library, wiringPi.h is the library for programming Raspberry Pi's GPIO pins, and thread library is the C++ standard library for creating threads.

Next, the program defines several constants, including the GPIO pin code of the soil moisture sensor, the GPIO pin code of the water pump, the threshold range for soil moisture measurement, etc. Among them, the threshold range is used to control the switch of the water pump. When the soil moisture is below the lower threshold, the program will automatically turn on the water pump for watering; when the soil moisture is above the upper threshold, the program will turn off the water pump to avoid over-watering.

Then, the program defines a function named "watering" to control the switch of the water pump. The function takes a digital signal representing the soil moisture value as a parameter, and the function will judge whether the soil moisture is too low or too high according to the value of the digital signal and control the switch of the water pump accordingly.

In the main function of the program, the program first calls the "wiringPiSetup()" function to initialize the wiringPi library. If the initialization fails, an error message will be output, and the program will exit. Then, the program sets the soil moisture detection pin and the water pump control pin to input mode and output mode, respectively. Next, the program enters an infinite loop, reads the soil moisture value, and passes it to the "watering" function for processing. At the same time, the program also outputs the current soil moisture value and the digital signal value information.

To avoid blocking the main thread, the program uses the thread library to create a thread, with the "watering" function as the processing function and passes the "soil_moisture_digital_value" parameter to the function. The thread is detached using the detach() function, allowing it to run in the background. This way, while reading the soil moisture value in the main thread, the water pump switch can also be controlled.

Finally, the program uses the delay function to delay for one second to avoid frequent reading of soil moisture value and water pump switch status and leaves enough time for thread event processing. The program runs in an infinite loop until it is manually stopped.

4. Intelligent Car System

The intelligent car is a robot based on embedded systems, consisting of hardware components such as sensors, chips, and power execution structures, each with different functions and roles.

4.1 Hardware components

Sensors are the perception module of an intelligent car, providing necessary information by sensing and monitoring the surrounding environment. The ultrasonic sensor can measure the distance to an object and is used for obstacle avoidance and collision avoidance. The photosensitive sensor can measure the light intensity and is used to detect lighting conditions. The camera sensor can capture images and videos and is used for identifying and tracking target objects. The output signals of these sensors can interact and coordinate with other parts such as chips and power execution structures to provide necessary information for the movement of the intelligent car.

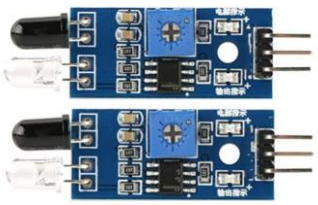

Fig 4.1.1. Infrared sensor

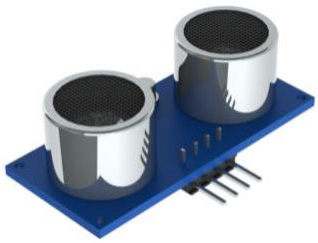

Fig 4.1.2. Ultrasonic sensor

The chip is the control module of the intelligent car, controlling power execution structures such as servo motors and motor drivers to enable the intelligent car to move and turn. The PCA9685 is a PWM controller used to control the speed and direction of servo motors and motor drivers, thereby controlling the movement of the intelligent car. The PCF8591 is an analog-to-digital converter used to convert the analog signals output by sensors to digital signals for the chip to process and interpret.

Fig 4.1.3. PCA9685

The chips can interact and connect with other parts such as sensors and power execution structures. Through the processing and interpretation of data, the intelligent car can be intelligently controlled and managed. The power execution structure is the power output module of the intelligent car, including servo motors and motor drivers. The servo motor is a high-precision motor that can adjust the speed and angle according to the input PWM signal, thereby controlling the direction and movement of the intelligent car. The motor driver is an electronic circuit module used to convert electrical signals to mechanical motion, controlling the speed and power of an intelligent car. The control signals of these power execution structures can be issued by the chip and adjusted and coordinated through the feedback signals of the sensors to achieve accurate control and smooth operation of the intelligent car.

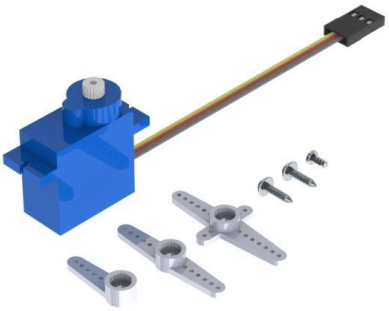

Fig 4.1.4. servo

Fig 4.1.5. motor

4.1 Smart car function

4.2.1 motion control function

1）Motor

By controlling the motor's speed and direction through the PCA9685 and Raspberry Pi, the speed of the four motors can be adjusted individually, thereby controlling the movement of the four wheels. This ultimately enables the car to move forward, backward, turn left, and turn right. Moreover, through actual testing of the influencing factors (constants), precise control of the car's movement speed, distance, and turning angle can be achieved. (The influencing factors obtained through actual testing and calculation are affected by the condition of the venue's surface and battery power).

The Raspberry Pi uses I2C to drive the 16-channel PCA9685. Channels 0-7 control the forward and backward movement of the four wheels, respectively. After initializing the PCA9685, a write operation is performed on channel 0. By writing the duty cycle of the PWM wave, the voltage value is adjusted, thereby controlling the wheel's speed.

We then construct functions to control the speed of the four wheels individually and encapsulate this part into a Motor class for controlling the motor. The subsequent method calls are as follows:

```
Motor m1 = Motor(fd);
m1.MotorGo(2000,2000,2000,2000, movingTime)
```

2）Servo

We use servos to implement the "head swing" of the car. Through the head swing, the car can detect obstacles in different directions. The Raspberry Pi controls the servos through the PCA9685. There are two servos, one for controlling left and right turning, and the other for controlling up and down turning. The two servos are connected to channels 8 and 9 of the PCA9685. After determining the servos to control, we input the predetermined angle and calculate the difference with the current angle. Then, we calculate the pulse needed based on the difference, and the PCA9685 controls the pulse duration, causing the servo to rotate a certain amount. Finally, the servo rotates to the specified angle. Define the servo control as class Servo. i.e.

```
Servo servo=Servo(fd);
servo.setServo(0,180)
```

By calling the Servo class through the code, we introduce the servo instance and then use the servo with channel 0 to rotate its angle to 90 degrees.

4.2.2 Obstacle avoidance

First, we use ultrasonic distance measurement. The distance measurement method records the time difference between the ultrasonic wave being emitted, reflected, and collected by the ultrasonic sensor. The speed of sound is then introduced to calculate the distance of the obstacle ahead. The function of using ultrasonic distance measurement is defined as class Ultrasonic.

```
digitalWrite(Trigger_pin,HIGH); // make trigger_pin output 10us HIGH level ,
delay(0.00001);
digitalWrite(Trigger_pin,LOW);
int pulseTime = pulseIn()* 340.0 / 2.0 / 10000.0;
distance_cm[i] = pulseTime;
```

We call the motor class in section 4.2.1 to control the movement of the car, call the Ultrasonic class to obtain the distance between the car and the obstacles, and call the Servo class to control the direction of the ultrasonic sensor. This allows for the detection of distances in three directions: front, left, and right. Based on the measured distances in these three directions, the car can choose its next movement operation, ultimately achieving obstacle avoidance.

4.2.3 Path planning and execution

1）Path planning

In path planning, we first measure the terrain and simplify the map by abstracting obstacles as circles. Then, we set the starting point and destination. Finally, we apply the Potential Function method for path planning. Taking a living room scenario as an example, we include two sofas and four table legs in the map. The optimal path obtained is as follows.

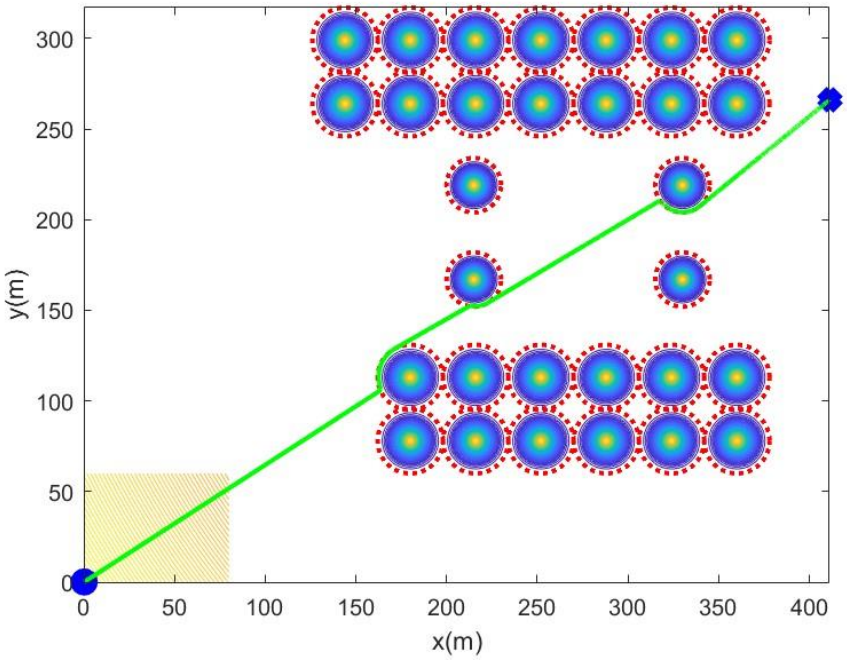

Fig 4.2.3. Optimal path in room

The principle of the Potential Function method is to set the force of the destination on the car as an attractive force while setting the force of obstacles on the car as repulsive forces. By setting the attractive and repulsive forces, the car can ultimately reach its destination from the starting point.

2）Exercise execution

After path planning, we obtain a series of path points. These points form the optimal path. However, the points required for completing the path planning image are far more than the points needed for the car's actual movement. Therefore, we select an appropriate number of points from the optimal path points as the car's path points, connecting them to form the real path. Finally, we issue commands to the car to move along the real path.

It is worth mentioning that, due to the small size of the car, it is greatly affected by the terrain. When the terrain has different friction forces and inclinations, the car's driving distance and turning angle will deviate. We set this influence as variables K and K_angle and include them in the commands for controlling the motor. When changing the terrain, we only need to measure and assign the corresponding values of K and K_angle to make the car's guidance function adaptable to different terrains.

In this function, we call the Motor, Ultrasonic, and Servo classes, enabling automatic path planning and movement of the car after determining the starting point and destination in the field.

4.2.4 Start Function

1） Infrared sensor activation

We use an infrared sensor that sends a low-level signal to the Raspberry Pi when it detects a nearby living being. Upon receiving the signal, the Raspberry Pi activates subsequent functions. The front infrared sensor uses pin 33 of the Raspberry Pi, and the rear infrared sensor uses pin 32. The functions getFrontValue() and getBackValue() for reading the signals are placed in a class.

2）Light-sensitive sensor

The light-sensitive sensor converts the light intensity signal into a voltage signal through the digital-to-analog converter PCF8591 and sends it to the Raspberry Pi. Different functions are activated based on the range of light intensity values. The functions for reading the light intensity signal values, getLeftValue() and getRightValue(), are placed in the Light class. The A0 port corresponds to the left light-sensitive sensor, and the A1 port corresponds to the right light-sensitive sensor. 。

```
front infrared: 1--back infrared: 1--light_left: 1313.73--light_right: 1313.73
front infrared: 1--back infrared: 1--light_left: 2598.04--light_right: 2607.84
front infrared: 1--back infrared: 1--light_left: 2076.47--light_right: 1974.51
front infrared: 1--back infrared: 1--light_left: 2364.71--light_right: 2368.63
go home
go to garden
front infrared: 1--back infrared: 1--light_left: 1313.73--light_right: 1313.73
front infrared: 1--back infrared: 1--light_left: 1313.73--light_right: 1313.73
front infrared: 1--back infrared: 1--light_left: 1294.12--light_right: 1294.12
front infrared: 1--back infrared: 1--light_left: 1296.08--light_right: 1296.08
front infrared: 1--back infrared: 1--light_left: 1294.12--light_right: 1294.12
front infrared: 1--back infrared: 1--light_left: 1274.51--light_right: 1274.51
front infrared: 0--back infrared: 0--light_left: 1254.9--light_right: 1254.9
front infrared: 0--back infrared: 0--light_left: 1274.51--light_right: 1274.51
front infrared: 0--back infrared: 0--light_left: 1274.51--light_right: 1274.51
go home
```

Fig 4.2.4. The output of the infrared sensor and the light sensor values and the corresponding functionality activation.

University of Glasgow

4.3 Function concatenation and overall performance

After implementing the above functions and encapsulating them using classes, we have achieved the automatic "leave home" and "return home" functions for the car.

The function description is as follows: When the light-sensitive sensor measures a light intensity value greater than 500, it indicates that the sun has risen in the morning. The car then activates its "leave home" function, performs path planning, and controls the car to move from inside the house to a designated location in the garden. If the light intensity is too strong, indicating that the harsh midday sun is harmful to plants, the car activates the path planning and "return home" functions to avoid the intense sunlight. After waiting for a certain period, when the sunlight reaches an appropriate range, the car leaves home and enters the garden again. Another trigger for the "return home" function is when the infrared sensor detects the presence of a living being nearby. This indicates that a living might be trying to steal vegetables or eat plants, so the car will also activate its "return home" function.

During the "leave home" and "return home" processes, we also added obstacle avoidance functionality in path planning and movement. Specifically, path planning targets obstacles that are already included in the map, while obstacle avoidance targets sudden obstacles that are not on the map. If the ultrasonic sensor detects an obstacle on the real path, the car will first bypass the obstacle and then discard some points on the real path, continuing to move along the remaining real path. This combines static obstacle avoidance in path planning with dynamic obstacle avoidance, achieving a more comprehensive automatic operation function for the car.

5. Overall System Assembly

To seamlessly integrate the irrigation and car systems, a customized plastic structure has been designed, as depicted in the accompanying drawing. This innovative structure comprises two distinct components - the upper part houses the soil and plants, while a perforated plate facilitates effortless drainage. The water pump is conveniently located in the immediate vicinity. Meanwhile, the lower part of the structure provides ample support for the car system, effectively distributing the weight from the upper portion while safeguarding the system's hardware.

However, because the ultrasonic module, which cannot be obscured, was overlooked in the preliminary design, the lower part had to be moved in practice, so that the upper and lower parts are not perfectly aligned, as shown in the diagram.

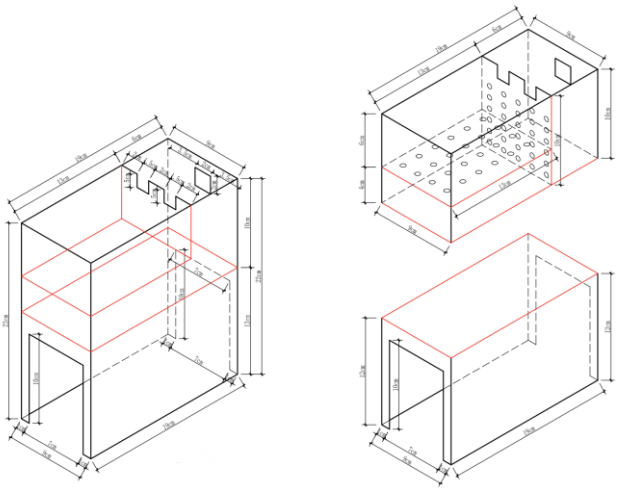

Fig 5.1. Construction of the car frame structure

Fig 5.2. Finished System

6. System Test:

Test the function and performance of the system, including the irrigation effect, automatic driving function, image recognition accuracy, etc.

7.Conclusion and Outlook:

One of the drawbacks of the irrigation system is the lack of devices with ADC (Analog-to-Digital Converter) functionality, which means that the Raspberry Pi program can only read binary digital signals instead of detailed values. For intelligent control, this aspect needs further in-depth research. More sensors can be added to achieve more comprehensive monitoring and protection, such as nitrogen, phosphorus, and potassium nutrient element sensors, to automatically detect the content of various nutrients in the soil and add an appropriate amount of fertilizer based on the growth needs of different plants, thus promoting plant growth.

The future outlook for intelligent irrigation systems involves using multiple sensors, such as soil moisture sensors, weather sensors, temperature sensors, and water quality sensors. These sensors can monitor environmental parameters and plant growth status in real-time, sending data to the central controller. The central controller must have data processing, decision-making, and control capabilities to process sensor data in real-time, make decisions, and control operations based on preset algorithms and strategies, thereby achieving automated and intelligent irrigation control. By employing Internet of Things（IoT）technology, sensors and central controllers can be connected to the cloud or other remote devices to enable remote monitoring and control. For instance, IoT platforms and cloud computing services can be used for data storage, analysis, and processing, while smartphones and computers can serve as endpoints for remote monitoring and control. And finally, intelligent irrigation systems need to be equipped with actuator devices, such as water pumps, irrigation pipelines, and sprinkler heads. These devices can be turned on or off according to the control signals from the central controller, achieving automated irrigation operations.