Smart Aquarium Care System

Project Report

Abstract

The primary objective of this project is to design and implement an automated fish feeding system for an aquarium that operates without human involvement. This project utilizes the Atmega32 microcontroller, GSM module, DHT11 sensor, servo motor, and an LCD display to create a fully automated fish-feeding system. The LCD display serves as an informative interface, showing essential data such as the time passed since the last fish feeding and the count of feeding gate activations, providing users with a comprehensive view of the system's performance. Additionally, buttons are integrated into the system, allowing users to conveniently adjust parameters based on individual user preferences and specific aquatic pet care requirements. The system ensures that fish are fed regularly and monitors and provides real-time information about the ambient temperature around the water level, ensuring optimal conditions for the fish's well-being. In case of unfavourable temperature conditions, it proactively sends SMS notifications to the user, allowing them to take immediate action and maintain a healthy environment.

Contents

1. Introduction

The "Smart Aquarium Care System" is mainly focused to use to maintaining aquariums, but it is also very suitable for home usage in small fish tanks. The system encompasses several key functionalities, with a primary focus on automated fish feeding according to predefined schedules. Moreover, it offers a user-friendly feature of sending SMS notifications to keep users informed and alert via text alerts.

1.1 Automated fish feeding

Users have the flexibility to define the intervals at which fish feed is dispensed. This feature is achieved by configuring a feeding schedule through the user interface, which is displayed on the LCD. After the welcome screen, users have the option to customize their feeding schedule. Using Button 1, users can select from predetermined time intervals, with options including 4, 6, and 8-hour intervals. Button 2 allows users to proceed to the next screen.

In the next screen, the first row displays the number of times the food gate opens for food dispensing. This value can be customized based on the number of fish and their feeding requirements, allowing users to define it as needed. The second row indicates the time elapsed since the last food dispensing. Users can adjust the number of times food is dispensed by clicking button 1, with the option to set it between 1 to 10. Button 2 provides the ability to stop the timer (pause). Pausing the timer enables users to manually dispense food as needed, and when the button is pressed again, the timer will resume the automatic feeding process.

In addition to these buttons, there is another critical button used to reset the system. It is essential to note that when clicking the reset button, the user must ensure that the entire food container is filled. Failure to do so may result in a low food level message, even if the container contains food.

1.2 Sending SMS Notifications

The system sends SMS notifications to the user in two specific situations. The first scenario is when the temperature is outside the optimal range for the fish. The second situation occurs when the food container's food level is low.

To achieve the functionalities described above, the system utilizes the Atmega32 microcontroller in combination with several components, each serving a specific purpose in the overall functionality of the system. The components used and their roles in processing the above functionalities are outlined below:

- **Atmega32 Microcontroller**: The central processing unit of the system, responsible for executing the programmed instructions and controlling all other components.

- **GSM Module**: This component enables communication with the user through SMS notifications, providing alerts about temperature and food levels.

- **The DHT11 senso**r: Measures temperature and humidity around the fish tank, ensuring that the environmental conditions are suitable for the fish.

- **Servo Motor**: The servo motor controls the fish feeding gate, allowing the system to dispense food at scheduled intervals.

- **LCD Display**: An integrated LCD screen provides real-time information to the user, including the time elapsed since the last fish feed and the number of times the fish feeding gate has operated.

- **User Interface Buttons**: These buttons allow the user to customize the feeding schedule, stop the timer, and reset the system if necessary.

- **Power supply**: Provides the necessary power to operate the entire system.

2. Methodology

The programming of the Atmega32 microcontroller was carried out using the Programmer's Notepad software, which provides a user-friendly environment for coding and development on the Windows operating system platform. Another alternative method for programming on Windows is through the Atmel Studio software. It is important to note that when using the Windows operating system, the installation of USBasp drivers is an essential step.

The correct interface between the USBasp and the Atmega32 microcontroller is a vital step. A loose connection can lead to failures in uploading the code to the microcontroller.

Figure 1: Physical appearances of USBasp
Source by: https://images.app.goo.gl/JrrTuW7php1gcpqN7

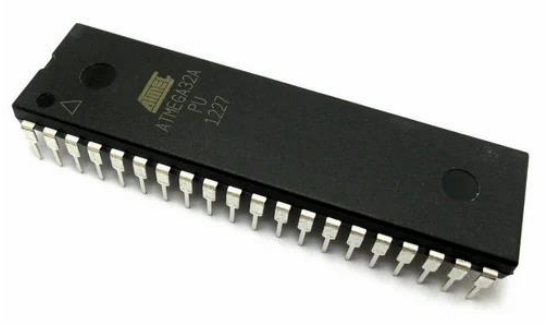

Figure 2 : Physical appearances of Atmega32
Source by: https://images.app.goo.gl/gvFu1i1yRmRBLLt56

As a first step, the microcontroller was connected to the breadboard, and a simple code was executed to verify the connection between the USBasp and the microcontroller. Subsequently, the components were individually connected to the microcontroller, and specific code was developed for each component.

After developing codes for each individual component needed for the project, all these separate codes were integrated into a single code file (main code). Following this integration, any errors in the main code were identified and corrected.

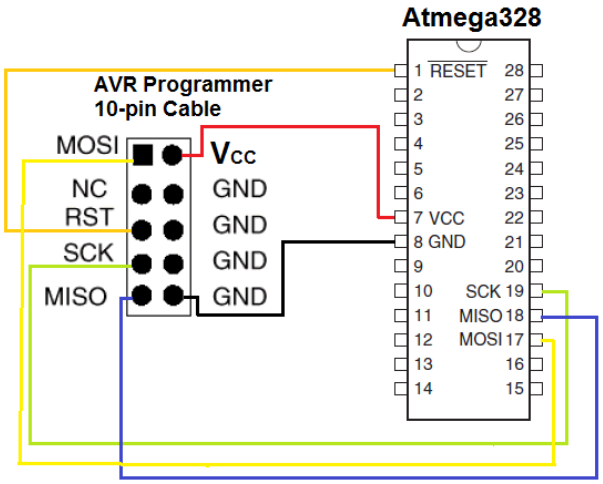

Figure 3: Pinout diagrams of Atmega32 and USBasp
Source by: https://images.app.goo.gl/Lt9UtecvgsRcLMnHA

Below is a list of main components used with the Atmega32 in the project, along with information about their usage and interfaces:

1. **16 × 2 Liquid Crystal Displays (LCD)**

Figure 4: Physical Appearance of the LCD Display
Source by: https://images.app.goo.gl/A9Xx2WuKsBd8N9q17

The LCD is operated in 4-bit mode, with connections established to the Atmega32 pins PC0, PC1, PD4, PD5, PD6, and PD7. Additionally, a 10k-ohm variable resistor was linked to the LCD to regulate display brightness.

2. Servo motor

Figure 5: Physical Appearance of the Servo Motor(5g)
Source by: https://images.app.goo.gl/MRee89oN2DRDunVF9

The servo motor was employed to rotate the feeding mechanism within a defined angle. A 9g servo motor was selected for this purpose. Its signal pin was connected to PD5(OC1A) on the Atmega32 microcontroller.

3. DHT11 - Temperature and humidity sensor

Figure 6: Physical Appearance of the DHT11 Sensor
Source by: https://images.app.goo.gl/AAQLyhYo9RgwCnhf9

Figure 7: The DHT11 sensor was utilized to monitor the temperature in the fish tank. It provides a 40-bit data stream containing temperature and humidity information, including Integral Temperature (integer part), Decimal Temperature, Integral Humidity (integer).

4. GSM Module – SIM800L

Figure 8: Physical Appearance of the SIM800L Module
Source by: https://images.app.goo.gl/yfY9MN7EmVVxeyXZ6

Unlike other components, it operates at 3.7 V, so a diode was employed to reduce the voltage to the required level. When connecting the GSM module to the Atmega32 microcontroller, it is important to make sure to link the TXD (Transmit Data) pin of one to the RXD (Receive Data) pin of the other and vice versa. This is important for proper communication using UART (Universal Asynchronous Receiver/Transmitter).

After ensuring that all the components worked correctly with the main code, the focus shifted to designing the PCB. The breadboard setup was used for testing the code. The "Eagle" software was used to design the PCB. The process started with making a schematic diagram and then moving on to the board diagram.

It is important to consider the board size and allocate sufficient space for each component. Insufficient space on the board can make it hard to attach the devices.

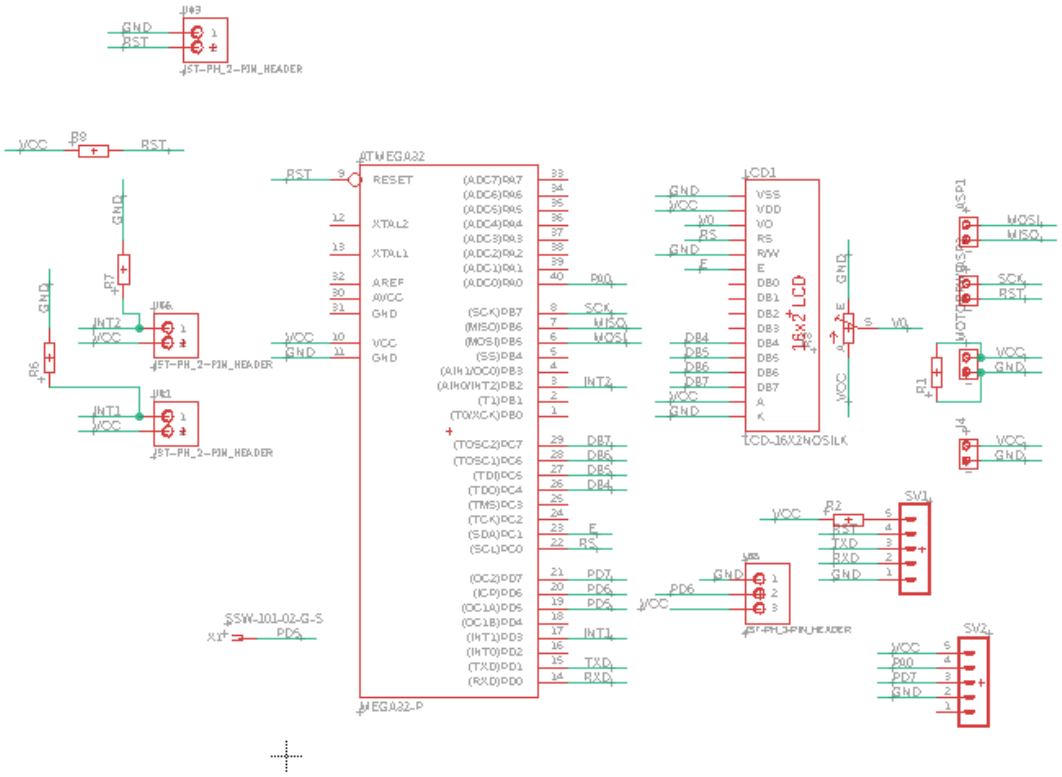

Figure 9: Schematic Diagram of the Circuit

Two PCBs were utilized in this project. One PCB was responsible for connecting all the components to the Atmega32 (Main PCB), while the other PCB powered the main PCB and provided a separate power line for the servo motor. The power supply used was a 12V, 1A power supply, which was reduced to 5V using a step-down converter.

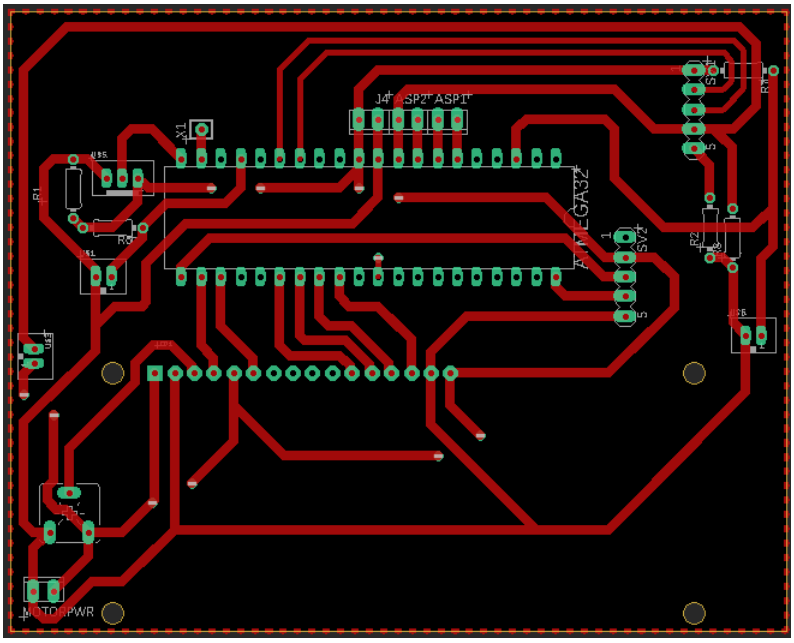

Figure 10: The PCB designed was made using Eagle software

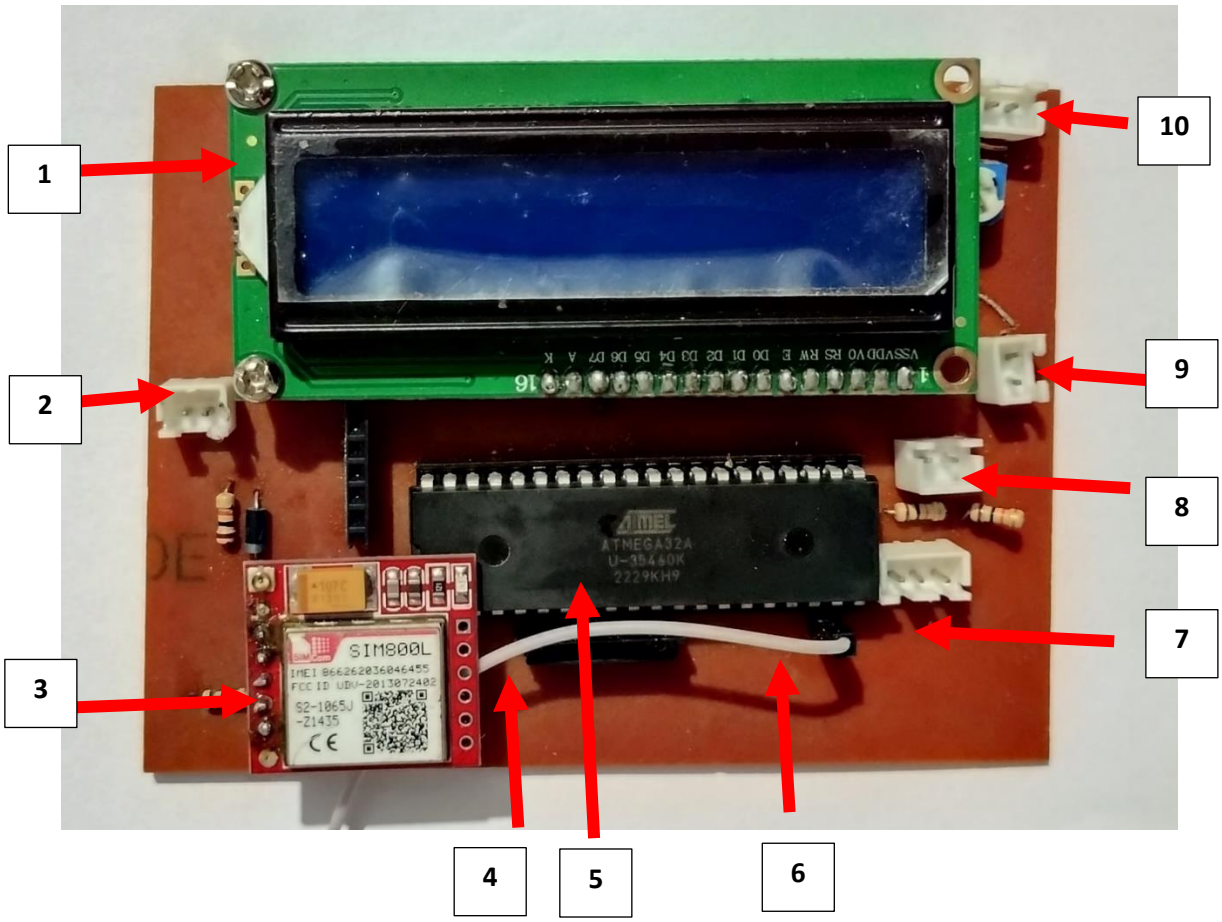

Figure 11: **(1) LCD display 16×2, (2) Button 1, (3) GSM 800L module, (4) USBasp heads, (5) Atmega32 microcontroller, (6) Servo motor data wire, (7) DHT 11 sensor pin heads, (8) Button 2, (9) Reset Button, (10) Power input.**

3. Discussion

The GSM module encountered several issues throughout the project, often arising from insufficient power. It is essential to provide sufficient power to the module for it to function correctly.

The SIM800L module features an LED that signifies the status of your cellular network. Depending on its current state, the LED will blink at various rates. There are primarily three modes.

Figure 12: GSM Module Connection Modes
Source by: https://lastminuteengineers.com/sim800l-gsm-module-arduino-tutorial

In this project, the third state is essential, as it involves establishing a connection with the cellular network for sending and receiving SMS. To verify the correct network connection of the device, you can simply provide power to the GSM module and observe whether the LED begins blinking after approximately three seconds.

Depending on the state, the module requires varying power levels. The maximum peak current occurs when it is in "transmission burst" mode, demanding up to 2 A of peak current. Typical jumper wires often cannot supply such high peak current, and if the device does not receive the power it requires, it can result in connection issues to the network and may even reset the entire Atmega32 program.

Even with a proper connection, the GSM module may encounter connectivity issues due to a lack of signal strength. In such cases, the best approach is to change the SIM card (to a different Service Provider) and attempt to establish a connection once more.

The servo motor is another high-power consuming device. An additional power source was supplied to it. Failure to do so can lead to flickering in the LCD display and may even damage the Atmega32 chip.

Both external and internal interrupts were employed in the system. Internal interrupts were used for the timer responsible for establishing the feeding schedule. The external interrupts were assigned to two buttons (Button 1 and Button 2) and were connected to the INT1 and INT2 ports on the Atmega32 microcontroller.

4. Results

The device performed as anticipated, with most challenges arising from the GSM module. Connecting to the cellular network can be a bit challenging due to weak network signals and occasional power issues.

All the other components used with the Atmega32 were found to function seamlessly together, demonstrating compatibility and smooth operation of the system.

5. Conclusion

The "Smart Aquarium Care System" represents a significant step forward in aquarium management, with features such as automated feeding, real-time environmental monitoring, and SMS notifications. Operating this device is very straightforward. The provided user guide offers a comprehensive overview of how to effectively use the device, ensuring a user-friendly experience.

The Smart Aquarium Care System is designed to cater to small aquariums and home users, and its potential for improvement is promising. Future enhancements could include features like water quality monitoring and light control for the fish. Additionally, the system can be modified to incorporate water filtering capabilities for maintaining water cleanliness.

6. References

Last Minute Engineers (2018). *Send Receive SMS & Call with SIM800L GSM Module & Arduino.* [online] Last Minute Engineers. Available at: https://lastminuteengineers.com/sim800l-gsm-module-arduino-tutorial.

Arduino Forum. (2021). *GSM issue SIM800L*. [online] Available at: https://forum.arduino.cc/t/gsm-issue-sim800l/689996.

www.electronicwings.com. (n.d.). *Interfacing LCD16x2 with AVR ATmega16/ATmega32 in 4-bit mode | AV.*. [online] Available at: https://www.electronicwings.com/avr-atmega/interfacing-lcd-16x2-in-4-bit-mode-with-atmega-16-32-.

www.electronicwings.com. (n.d.). *DHT11 Sensor Interfacing with AVR ATmega16/ATmega32. | AVR ATmega.*. [online] Available at: https://www.electronicwings.com/avr-atmega/dht11-sensor-interfacing-with-atmega16-32.

7. Appendix

Code for the GSM module.

```
#define F_CPU 8000000UL // Define the CPU clock frequency as 8 MHz

#include <avr/io.h> //Include AVR library for I/O and register definitions
#include <util/delay.h> //Include AVR library for delay functions
#include <avr/interrupt.h> //Include AVR library for interrupt handling

// Define UART baud rate pre-scale
#define BAUD_PRESCALE (((F_CPU / (USART_BAUDRATE * 16UL))) - 1)

// Initialize UART communication with the specified baud rate
void UART_init(long USART_BAUDRATE);

// Receive a character from UART
unsigned char UART_RxChar(void);

// Transmit a character through UART
void UART_TxChar(char ch);

// Send a string through UART
void UART_SendString(char *str);

int main(void) {
    while (1) {
        // Initialize UART communication with a baud rate of 9600
        UART_init(9600);

        // Send AT command to check the module's response
        UART_SendString("AT\r\n");
        _delay_ms(2000);

        // Disable command echo
        UART_SendString("ATE0\r\n");
        _delay_ms(2000);

        // Set SMS text mode
        UART_SendString("AT+CMGF=1\r\n");
        _delay_ms(2000);

        // Send an SMS to the specified number
        UART_SendString("AT+CMGS=\"+xxxxxxxxx\"\r\n");
        _delay_ms(2000);

        // Send the SMS message: "Food Level Low"
        UART_SendString("Food Level Low ");

        UART_TxChar(26);

        // Wait for 10 seconds before repeating the process
        _delay_ms(10000);
    }
    return 0;
}
```