EMBEDDED SYSTEMS

Project

Master of Science Robotics Engineering

Group
Ankur Kohli (5160903)
Ammar Iqbal (5183355)
Basit Akram (5161322)

Submitted to
Prof. ENRICO SIMETTI

Dipartimento di Informatica, Bioingegneria, Robotica e
Ingegneria dei Sistemi (DIBRIS)
Universit'a degli Studi di Genova (UniGe)

Preface

The work is based on Embedded Systems which involves the implementation of timers, interrupts, SPI, UART, parser ADC, PWM, Scheduling, and so on. This assignment deals with MPLAB IDE Software, XC16 Compiler, HTerm serial software and Microchip Microcontroller Board.

Acknowledgements

We take this opportunity to express our sincere thanks to *Prof. ENRICO SIMETTI* for his guidance and support. We would also like to express our gratitude towards him for showing confidence in us. It was a privilege to have a great experience working under him in a cordial environment. We are very much thankful to the *University of Genoa*, for providing us with the opportunity to pursue *M.Sc in Robotics Engineering* in a peaceful environment with ample resources. In the end, we would like to acknowledge our parents and friends. Without their support, this work would not have been possible.

Abstract

This project report is about the ***Implementing a basic control System for an Autonomous Underwater Glider*** *in which the of* ***timers, interrupts, UART, PWM, SPI, parser, Scheduling, and so on*** *to determine about the operations performance. The specific goal, in this case, is to know the implementation of operations with real-time hardware. Also,* ***case study of a real-time control system****. This report brings light to* ***Embedded Systems*** *for* ***operations****. Furthermore, the purpose of this report is to provide the approaches used during the development of code and implementation on microcontroller board. For this assignment,* ***MPLAB IDE Software, XC16 Compiler and Microchip Microcontroller Board, HTerm serial software*** *is used and Embedded C programming is a platform for developing code.*

Contents

1 Introduction

A microprocessor-based computer system with software that is intended to carry out a specific task, either independently or as a component of a larger system, is known as an ***embedded system***. An integrated circuit built to perform computing for real-time processes is at the heart of the system.

From a single microcontroller to a group of connected processors with networks and peripherals, complexity can range from having no user interface to having intricate graphical user interfaces. Depending on the task for which it is created, an embedded system's complexity varies greatly.

Applications for embedded systems include hybrid cars, avionics, digital watches, microwaves, and more. Embedded systems consume up to 98% of all produced microprocessors.

2 Methodology

In this project, our main task is to implement the operations such as ***timers, interrupts, UART, PWM, SPI, data-parser, Scheduling, and so on***. Also, this assignment deals with MPLAB IDE Software, XC16 Compiler, HTerm serial software and Microchip Microcontroller Board. We used ***dsPIC30F4011 Enhanced Flash 16-bit Digital Signal Microcontroller Board*** as shown in figure 1, for the real-time implementation of the developed code.

Goal: A microcontroller board is connected to three motors. One is the main propeller, pushing the glider forward. The second motor actuates the rudder of the glider, allowing it to have a yaw motion. The third one is a linear motor that moves the battery pack, which in turn allows to statically change the pitch of the vehicle.

The microcontroller receives desired reference values for the forward speed, the pitch, and the rudder angle. These reference signals are sent through a serial interface. The microcontroller sends feedback messages back to the control PC to report a few status information.

Implementing a basic control system for an autonomous underwater glider involves several steps:

- **Connection to the Microcontroller Board:** The first step is to establish a communication link between the microcontroller board and the control PC. This can be done using a serial interface, such as USB, to transmit the reference signals and receive the feedback messages.

- **Reading Reference Signals:** The microcontroller should be programmed to receive the desired reference values for the forward speed, pitch, and rudder angle. These values should be stored in appropriate variables for use in the control algorithm.

- **Control Algorithm:** The control algorithm should determine the control signals sent to the motors based on the received reference signals and the feedback messages. For example, the algorithm can use a PID (Proportional-Integral-Derivative) control strategy to regulate the speed, pitch, and rudder angle.

- **Motor Control:** The microcontroller should be programmed to send the control signals to the motors. This can be done using PWM (Pulse-Width Modulation) signals or analog signals to control the motor speed and direction.

- **Feedback:** The microcontroller should send feedback messages back to the control PC to report the status of the system, such as the current speed, pitch, and rudder angle. This information can be used to adjust the reference signals and improve the performance of the control system.

- **Data Logging:** The microcontroller can also be programmed to log data, such as the reference signals, control signals, and feedback messages, to a file for later analysis.

- **Testing and Debugging:** Finally, the control system should be thoroughly tested and debugged to ensure that it operates as expected. This can be done by running the glider in various scenarios and verifying that the speed, pitch, and rudder angle remain within the desired limits.

Peripheral Features of the board:

- High current sink/source I/O pins: 25 mA/25 mA

- Timer module with programmable prescaler: Five 16-bit timers/counters; optionally pair 16-bit timers into 32-bit timer modules

- 16-bit Capture input functions
- 16-bit Compare/PWM output functions
- 3-wire SPI™ modules (supports 4 Frame modes)
- I2C™ module supports Multi-Master/Slave mode and 7-bit/10-bit addressing
- 2 UART modules with FIFO Buffers
- 1 CAN modules, 2.0B compliant

Source: dsPIC30F4011 Datasheet https://ww1.microchip.com/downloads/en/devicedoc/70135c.pdf

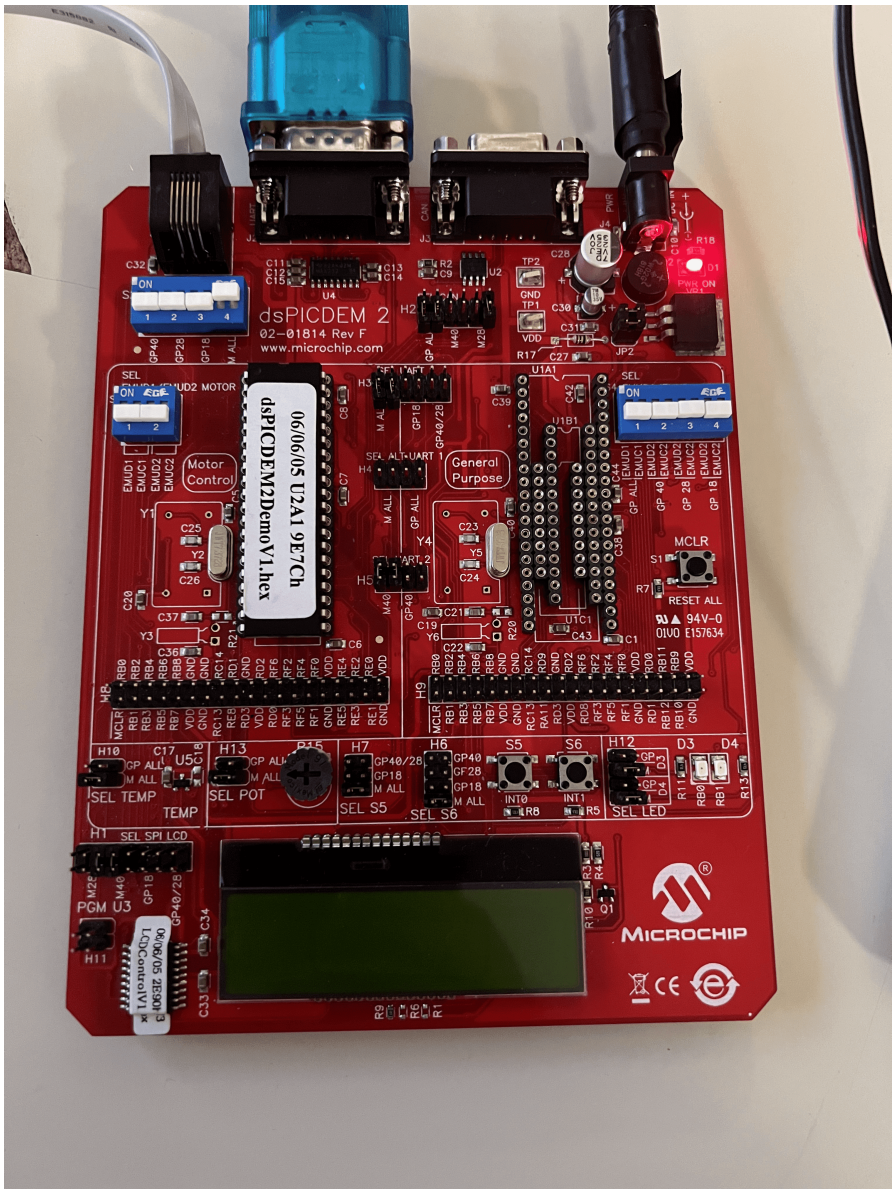

Figure 1: dsPIC30F4011 Board

DSP Engine Features:

- Dual data fetch
- Accumulator write back for DSP operations
- Modulo and Bit-Reversed Addressing modes
- Two, 40-bit wide accumulators with optional saturation logic
- 17-bit x 17-bit single cycle hardware fractional/ integer multiplier
- All DSP instructions single cycle
- ± 16-bit single cycle shift

Source: dsPIC30F4011 Datasheet https://ww1.microchip.com/downloads/en/devicedoc/70135c.pdf

Porgramm Burner: pickit3 Programmer

Figure 2, shows the program burner.

Figure 2: pickit3 Programmer

Hardware Specifications

These hardware specifications describe the properties and performance of three motors in a vehicle. The first motor has a maximum RPM range of -10000 to 10000, while the second and third motors have a maximum RPM range of -100 to 100. The motors are controlled using a PWM signal with a frequency of 1 kHz and a dead time of at least 3 microseconds to prevent issues with the H-bridge.

It's important to note that running the motors above their limits can cause damage and should be avoided. The duty cycle of the PWM signal corresponds to different RPM values for each motor, with 50% duty cycle corresponding to 0 RPM.

The forward speed of the vehicle can be approximated to be linearly proportional to the RPM of the first motor, with a maximum forward speed of 2 m/s when the first motor runs at 10000 RPM. The position of the battery pack affects the pitch of the vehicle, with a maximum pitch of 20 degrees at a position of 10 cm and 0 degrees at a position of 0 cm. The rotation speed of the rudder is proportional to the RPM of the third motor, with a maximum rotation speed of 5 deg/s and a maximum rudder angle range of -30 to 30 degrees.

Below are the hardware specifications:

- The main motor can run from -10000 to 10000 RPMs;
- The second and third motors can run from -100 to 100 RPMs.
- The motors are controlled through a PWM signal.
  - The frequency must be 1 kHz.
  - For motor 1, 50% duty cycle corresponds to 0 RPM, 0% corresponds to -11000 RPM and 100% corresponds to 11000 RPMs.
  - For motor 2 and 3, 50% duty cycle corresponds to 0 RPM, 0% corresponds to -110 RPM and 100% corresponds to 110 RPMs.
  - A dead time of at least 3 microseconds should be used to prevent problems with the H-bridge controlling the motors. o Running the motors above their limits can damage them and should be avoided.
- Ignoring dynamics, when motor 1 runs at 10000 RPMs the vehicle moves with a forward speed of 2 m/s. The speed is approximated to be linear with the RPMs.
- When motor 2 runs at 100 RPMs, the battery pack moves with 5 mm/s. The battery pack can move from position 0 to 10 cm. The pitch is linear with the position of the battery pack: again, ignoring dynamics, when the battery is at position 5 cm, the pitch is 0 degrees. When the battery is at position 10 cm, the pitch is +20 degrees, and when it is at 0 cm, the pitch is 0 degrees.

- When motor 3 runs at 100 RPMs, the rudder moves with 5 deg/s. The rudder can move from -30 to 30 deg.

Specification source from: https://2022.aulaweb.unige.it/mod/assign/view.php?id=87348

Firmware Requirements for the project

- The firmware should handle the cases where the reference exceeds the limits of the rudder angle and the battery pack position, and should automatically adjust the reference to the maximum/minimum limits if necessary.
- The firmware should handle any communication errors or failures between the control system and the PC, and should enter a safe mode in such cases, with all motor's velocity set to zero.
- The firmware should have a user-friendly interface for configuring and adjusting the parameters of the control system, such as the maximum and minimum limits for the rudder angle and battery pack position.
- The firmware should have robust error handling mechanisms, such as checks for divide-by-zero errors and buffer overflow errors, to ensure the reliable and safe operation of the control system.
- The firmware should be tested thoroughly for performance and functionality, including the ability to operate under different environmental conditions, such as high temperatures, humidity, and vibration.
- The firmware should be easily updatable so that future improvements and bug fixes can be implemented without requiring a hardware change.

The firmware requirements of the projects are given below:

1. The control systems should compute the desired RPMs of the three motors, given the reference speed and position (pitch and rudder angle).
2. The firmware must simulate the movement of the rudder and of the battery pack, using the aforementioned relationships between PWM and derivatives of rudder angle and battery pack position (integrating the velocities).

3. The control system must never generate PWM signals outside of the specifications of the motors.

4. The rudder angle and battery pack limits should never be exceeded.

5. If no references (i.e., the MCREF command) is received from the PC for more than 5 seconds, the firmware should enter a timeout mode:
   - All motor velocities should be set to zero.
   - Led D4 should blink at 5 Hz to signal timeout.
   - When a new reference is read, then the d D4 should stop blinking and commands should be given again to the motors.

6. The firmware must support receiving references at least at 10 Hz frequency (through a proper choice of baud rate).

7. The firmware must refresh the PWM values at least at 10 Hz frequency.

8. The firmware must send the PWM feedback message MCPWM at 5 Hz frequency.

9. The firmware must send the Position feedback message MCPOS at 10 Hz frequency.

10. The control system should blink led D3 at 1 Hz to always signal the correct functioning of the main loop, regardless of any state.

11. Given the chosen UART baud rate, the firmware should never lose a message due to its implementation (i.e., proper dimension of buffers), even with full use of the bandwidth.

12. The firmware should write on the LCD
    - First row: "Speed: x", where x is the desired forward speed
    - Second row: "R: n1", where n1 is the applied motor1 RPM (e.g. "R: 900")
    - If the button S6 is pressed, the data displayed on the LCD changes as follows:
      - First row: "R: x P: y", where x is the desired rudder angle, and P is the desired pitch angle
      - Second row: "R: x: P: y", where x is the actual rudder angle, and P is the actual pitch angle
    - If the button S6 is pressed again, the data displayed toggles again to the first one.

Firmware requirements source from: https://2022.aulaweb.unige.it/mod/assign/view.php?id=87348

UART Protocol:

Messages from the PC: $HLREF,speed,pitch,rudder* where speed (m/s) is the desired linear velocity for the robot, pitch (deg) is the desired pitch angle, and rudder (deg) is the desired rudder angle.

Messages to the PC:

- $MCPWM,n1,n2,n3* where n1, n2, n3 are the applied RPM.
- $MCPOS,p1,p2*, where p1 (deg) is the position of the rudder, and p2 (mm) is the position of the battery pack.

Tips for Writing to LCD with SPI

The following power-up sequence should be observed by the user's application firmware when writing characters to the LCD:

1. After any reset operation waits 1000 milliseconds to allow the LCD to begin normal operation. The cursor on the LCD will be positioned at the top row on the leftmost column.
2. Configure the SPI1 module on your dsPIC30F device to operate in 8-bit Master mode. The serial clock may be set for any frequency up to 1 MHz.
3. To write an ASCII character to the LCD at the location pointed to by the cursor, load the SPIBUF register with the ASCII character byte.
4. After the character is displayed on the LCD, the cursor is automatically relocated to the next position on the LCD.
5. To reposition the cursor to another column on any of the two rows, write the address of the desired location to the SPIBUF register. Addresses in the first row of the LCD range from 0x80 to 0x8F, while addresses on the second-row range from 0xC0 through 0xCF.

6. After 16 characters are written to the first row on the LCD, it is necessary for the user's application to write the address 0xC0 of the second row to the SPIBUF in order to roll the cursor over to the second row.

7. The user application must wait for a minimum of (8 bits / SPI Frequency) between writing two successive characters or addresses.

Source: Assignemnt pdf https://2022.aulaweb.unige.it/mod/assign/view.php?id=70551

2.1 Approach of the code

2.1.1 Implementation of Control System in the project

Firstly, we gave input as speed (in m/s) for the first motor. Since the relation between RPM and Speed is linear so we have directly mapped the speed on RPM. From this RPM, we computed the duty cycle which we gave to PWM Register i.e. PDC.

Secondly, we gave input as pitch (in mm) for the second motor. For the computation of RPM, we know the relation between pitch and position and position and RPM. So, the pitch we received and later we mapped the same pitch on position. Then we calculated the error between the desired and actual position and we calculated the action controller by multiplying the error with proportional gain. After these calculations, we mapped this value on RPM and again calculated the RPM. Finally, we calculated the actual position by adding the previous actual position with control action into a change of time (δt). From this motor, we are determining the position of the battery pack. Since the change in the position of the battery pack cannot be fast so, previously multiplied the error with the buffer gain.

Thirdly, last but not least this motor is used to control the rudder of the water glider. In this, we gave input as angle (in degree). Moreover, we directly mapped this angle on the RPM scale and through which we computed the RPM.

2.1.2 Implementation of States in the project

1. ***Control State:*** In this state, we implemented the aforementioned control system of the project. Through this control system, we calculated the RPMs for the three motors.

2. ***Safe State:*** In this state, it occurs when the input message is sent from the PC.

3. ***Timeout State:*** This state occurs when no data is sent from the PC for 5 seconds and reset all motor's RPM values to zero.

3 Results & Discussion

Figure 1 below shows that firmware will be written on LCD.

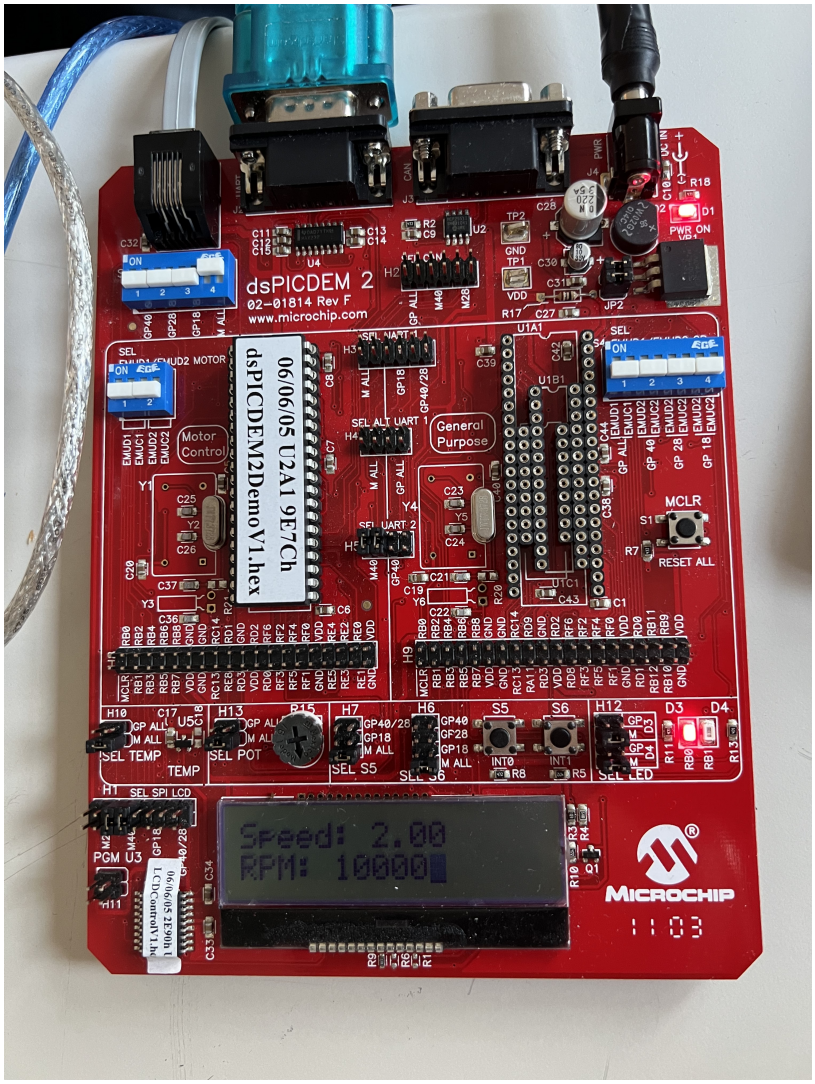

Figure 3: Button S6 interrupt is not pressed

Figure 1 shows when button S6 is not pressed. It shows the message as described below:

- First row: **"Speed: x"**, where x is the desired forward speed

- Second row: **"RPM: n1"**, where n1 is the applied motor1 RPM (e.g. in our "RPM: 10000")

Figure 2 shows when button S6 is pressed. If the button S6 is pressed, the data displayed on the LCD changes as follows:

- First row: **"R: x P: y"**, where x is the desired rudder angle, and P is the desired pitch angle
- Second row: **"R: x: P: y"**, where x is the actual rudder angle, and P is the actual pitch angle

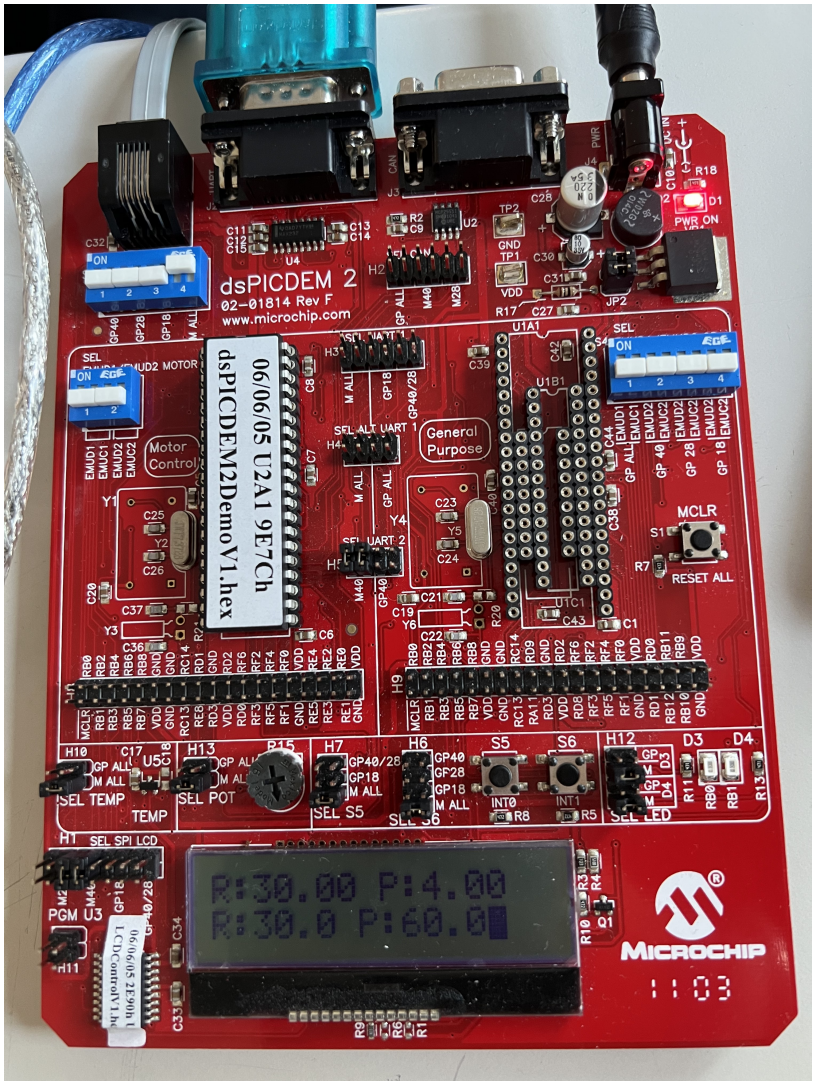

Figure 4: Button S6 interrupt is pressed

4 Conclusion

In this project, three motors are linked to a microcontroller board. The primary propeller, which moves the glider forward, is one. The second motor operates the glider's rudder, allowing it to yaw. The third one is a linear motor that moves the battery pack, allowing the vehicle's pitch to be changed statically.

The desired reference values for the forward speed, pitch, and rudder angle are sent to the microcontroller. Through a serial link, these reference signals are transmitted. The microcontroller sends feedback signals to the control PC in order to relay some status information.

The main objective of this project is to understand the concept of timers, interrupts, UART, SPI, parser, ADC, PWM, Scheduling and so on. By using the same we achieve the goal of the assigned task. We are able to design the ***Basic Control System for an Autonomous Underwater Glider*** as shown and discussed under Result & Discussion section.