Microcontrollers Final Report

Group: L10

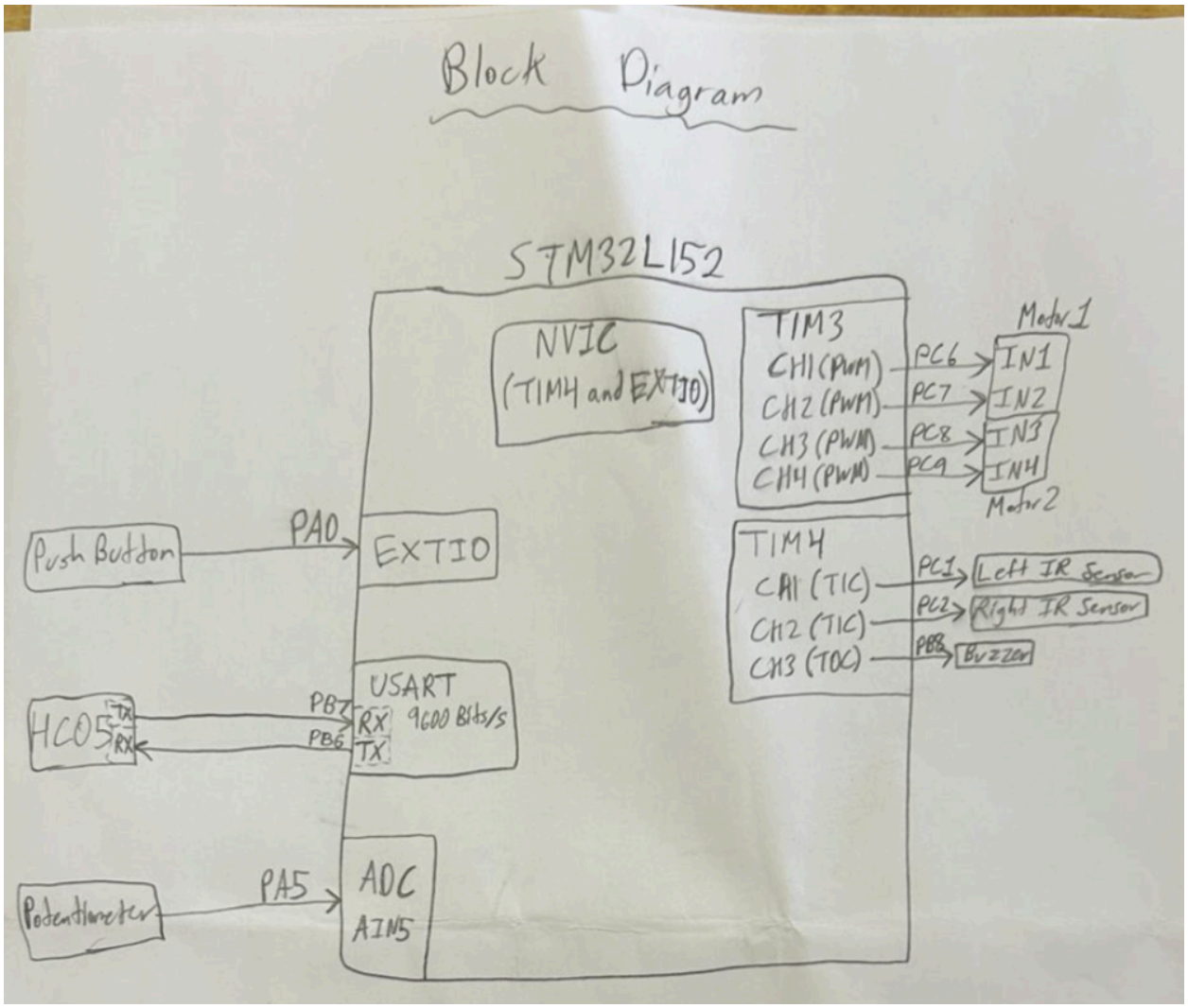
Members:

- **Jose Alberto: 100429289**
- **Ethan Anderson: 100532812**
- **Samuel Horowitz: 100529512**
- **Jarek Socha: 100529978**

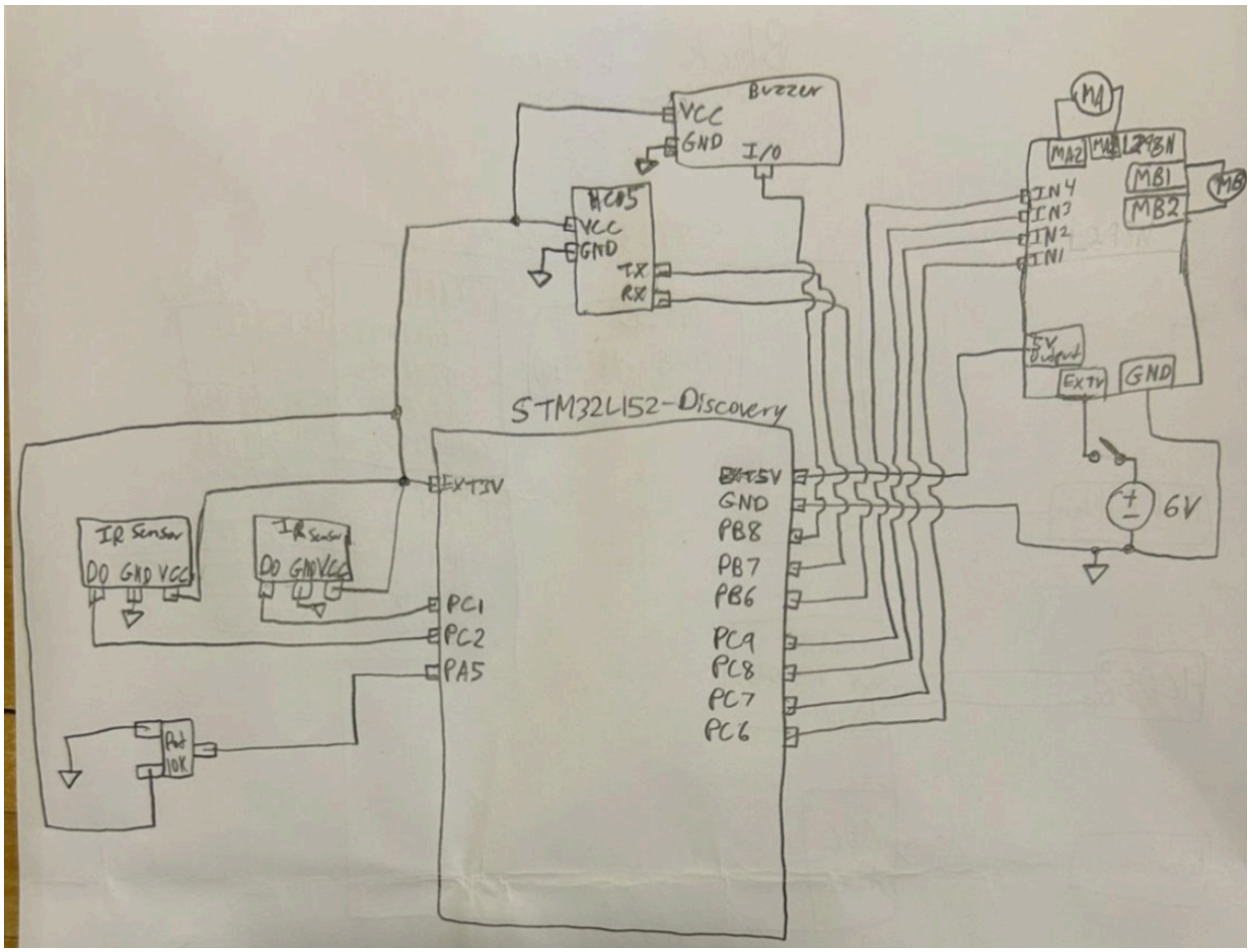
April 14th, 2024

Block Diagram: **3**
Schematics and Connections: **4**
Peripherals and Setup: **5**
Internal Peripherals 5
External Peripherals 6
IRQ and ISR Information: **8**
1. TIM4 Update and Capture Compare IRQ (TIM4_IRQHandler): 8
2. External Interrupt IRQ (EXTI0_IRQHandler): 8
3. UART Receive Complete IRQ (HAL_UART_RxCpltCallback): 9
ISR Flowcharts: **10**
TIM4_IRQHandler() Flowchart: 10
EXTI0_IRQHandler() Flowchart: 10
Conclusions: **11**

Block Diagram:



Schematics and Connections:



Peripherals and Setup:

The peripherals listed in this report were the ones utilized in the final development of the car. Although the STM32-L DISCOVERY board offers other peripheral options, some of which are integrated into the board, they were not employed in the development of this car and hence will not be listed.

Internal Peripherals

ADC:

- The ADC converted analog signals into digital values that were usable by the microcontroller. In this car the setup for the ADC began by designating the GPIO pin PA5 as an analog input. ADC registers were initially configured to deactivate the ADC. Then, overrun IRQ, scan mode, and EOC IRQ were disabled. The resolution was set to 12 bits, with a delay until data was read set to 1, EOC activated after each conversion, and continuous conversion mode enabled. Next, AIN5 was selected as the input channel. Finally, the ADC was activated, initiating the conversion process.

Timers (TIM3 and TIM4):

- The timers were utilized to generate PWM signals and measure time intervals to trigger events. TIM3 was configured for PWM signal generation. The prescaler was set to 48000-1, the counter was initialized to 0, ARR was adjusted to achieve a PWM of 100Hz with 10 steps, capture/compare was enabled for PWM compatibility, output was activated, and the update generation was triggered. Meanwhile, TIM4 was established as a basic timer without PWM. The prescaler was adjusted for approximately millisecond measurements, the counter was initialized to 0, ARR was set to its maximum value, and interrupts were enabled for capture/compare operations.

USART/ UART:

- The USART/UART facilitated communication between the microcontroller and external devices. In this lab, UART was set up by navigating to the IOC section of the STM IDE to initialize asynchronous mode and disable hardware flow control. Then the Baud Rate, Number of data bits, Parity, stop bit, communication direction, and oversampling were configured.

External Interrupts (EXTI):

- The External interrupts enabled the microcontroller to receive data about external events. In the case of this car, the EXTI was primarily configured for the user

button press. This was accomplished by enabling EXTI0, configuring it for falling edge input and IRQ, disabling the rising edge input, and linking it to PA0.

General Purpose Input Output pins (GPIO):

- The GPIO pins enabled the board to receive input or transmit output signals. The pins were assigned their respective use cases of Digital input, Digital output, Analog, or AF in the IOC and then utilized in the code. The configuration of the pins was predetermined in the instructions based on required functionality. PC 6-9 were designated as digital outputs for the motors, PC1&2 were allocated as digital inputs for the IR sensors, PB8 was reserved for the digital output of the buzzer, PA5 was configured as the analog input of the potentiometer, PB6 and PB7 were utilized for UART transmission and reception respectively, and PA0 was dedicated to the user button.

External Peripherals

Motors:

- Motor A received a connection to the motor controller between terminal 1 from OUT1 and terminal 2 from OUT2. To move forwards terminal 1 received a high connection and terminal 2 received a low connection. To move backwards terminal 1 received a low connection while terminal 2 received a high connection. If both terminals received a low connection the motor would stop. Motor B operated similarly, with its terminals connected to OUT3 and OUT4 respectively.

Motor Controller (L2985N):

- The motor controller received PWM signals from the discovery board and used them to dictate the power supplied to the motors. For motor A, the controller received a signal into IN1 from PC6 and IN2 from PC7 and processed it to supply power to the motor. If IN1 was high while IN2 was low, it controlled the motor to move forwards. While if IN1 was low and IN2 was high, it controlled the motor to move backwards. Finally if IN1 as well as IN2 was low, then the motor stopped. Motor B followed the same process, except it used IN3 from PC8 and IN4 from PC9. The car could also turn left and right using a combination of the two motors. By having motor A receive the signals to move forwards while motor B received the signals to stop the car would turn right. While if motor A received the signals to stop forwards while motor B received the signals to move forwards the car would turn left.

IR Sensors (TCRT5000):

- The IR sensors detect the characteristics of the object below it and communicate it to the discovery board. The left sensor will communicate to PC1 if it senses the object below it is white with a low signal or black with a high signal. The right sensor works the same way except for it sends its signal to PC2

Buzzer:

- The buzzer received a signal from the discovery board and emitted a sound when given a low signal. It received a PWM signal from the discovery board's TIM4 and buzzed at that given frequency.

Bluetooth Transceiver (HC-05):

- The bluetooth transceiver facilitated wireless communication between an external terminal and the discovery board. The Bluetooth transceiver received wireless Bluetooth communications from an external device, which it could relay to the discovery board's PB6 over its RX pin using the UART functionality. Additionally, it could receive data over a hard line using the UART function from the discovery board's PB7 pin to its TX pin, and then transmit the message over a Bluetooth connection to a terminal.

Potentiometer:

- The potentiometer enabled reception of variable resistance by the microcontroller. The variable voltage was provided to PA5, which had to be processed by the ADC in the microcontroller. That allowed the speed of the motor to be adjusted by an external source.

Push Button:

- The push button receives a physical press that is converted into a signal to the discovery board. The button connects to PA0 and triggers the EXTI on the falling edge which tells the car to start driving while in autonomous mode.

IRQ and ISR Information:

The code uses several IRQs (Interrupt Requests) and their corresponding Interrupt Service Routines (ISRs) for various functionalities:

1. TIM4 Update and Capture Compare IRQ (TIM4_IRQHandler):

Functionality:

- Delay Handler: The `TIM4_IRQHandler` IRQ is utilized to generate precise delays required for various tasks within the program. These delays are often crucial for coordinating different parts of the system or ensuring proper timing between actions.
- Buzzer Handler: By adjusting the timing parameters within the ISR, the frequency of the buzzer sound can be modulated according to the requirements of the application. This is used to generate different tones or patterns to convey specific information to the user.
- Sensors Handler: The `TIM4_IRQHandler` IRQ is used to periodically sample sensor data at regular intervals. This data can then be processed within the ISR to detect changes in sensor readings and take the correct actions.

Interrupt Source:

- This IRQ is triggered by events such as the overflow of the TIM4 timer and capture/compare events. Timer overflow occurs when the timer's counter register reaches its maximum value and rolls over to zero, generating an interrupt. Capture/compare events occur when the timer's counter value matches the values stored in the capture/compare registers, allowing for precise timing control and event capture.

2. External Interrupt IRQ (EXTI0_IRQHandler):

Functionality:

- User Button Press Event: The `EXTI0_IRQHandler` IRQ is used to handle events triggered by the press of a user button. This could involve initiating the autonomous mode in response to user input.

Interrupt Source:

- The `EXTI0_IRQHandler` IRQ is generated by a falling edge on the external interrupt line EXTI0. By connecting the user button to the EXTI0 line, the microcontroller can detect button presses and generate interrupts accordingly.

3. UART Receive Complete IRQ (HAL_UART_RxCpltCallback):

Functionality:

- UART Communication: UART (Universal Asynchronous Receiver/Transmitter) is used as a serial communication protocol. The `HAL_UART_RxCpltCallback` ISR is triggered whenever a byte is received via UART communication. This ISR is responsible for processing the received data and taking appropriate actions based on the received information.

Interrupt Source:

- The interrupt source for this ISR is the UART receive complete event. This event occurs when a complete byte of data is received and stored in the receive buffer of the UART peripheral. By handling this event in the ISR, the microcontroller can respond promptly to incoming data, ensuring smooth and efficient communication between different components of the system.

These IRQs and their corresponding ISRs play crucial roles in the overall functionality of the application, enabling tasks such as timing control, user interaction, and communication with external devices.

ISR Flowcharts:

We have two main ISRs in your code: TIM4_IRQHandler() and EXTI0_IRQHandler(). These ISRs handle timer interrupts and external interrupts respectively. Here's a general structure of how we could represent them in flowcharts:

TIM4_IRQHandler() Flowchart:

Start
Check if the delay interrupt flag is set.

- If yes:
    - Handle delay interrupt.
    - Start TIM3 for PWM motor control
    - Clear the delay interrupt flag.
- If no, move to the next step.

Check if the buzzer interrupt flag is set.

- If yes:
    - Handle buzzer interrupt.
    - Clear the buzzer interrupt flag.
- If no, move to the next step.

Check if the sensor interrupt flag is set.

- If yes:
    - Handle sensor interrupt.
    - Clear the sensor interrupt flag.
- If no, exit.

EXTI0_IRQHandler() Flowchart:

Start
Check if the EXTI0 interrupt flag is set.

- If yes:
    - Handle EXTI0 interrupt.
    - Clear the EXTI0 interrupt flag.
- If no, exit.

Conclusions:

The project successfully implements a robot with autonomous behavior, capable of moving forward, backward, turning left, and turning right. It also includes features like speed control and autonomous mode. Therefore, the functionality objective is achieved.

Hardware Setup: The microcontroller used is the STM32L152C Discovery board. GPIO pins are configured for motor control, sensor input, and other functions. Timers are utilized for generating PWM signals and managing delays.

Peripheral Usage:

- UART: Used for communication with an external device, presumably for receiving commands or sending status updates.
- Analog-to-Digital Converter (ADC): Configured for reading analog sensor values, likely for line following or obstacle detection.
- Timers (TIM3 and TIM4): Employed for generating PWM signals, timing delays, and handling periodic tasks through interrupts.
- External Interrupts (EXTI): Utilized for handling user input, such as pressing a button to trigger a specific action.
- General Purpose Input/ Output Pins (GPIO): Needed to receive and send signals to external devices.

Interrupt Handling:

- Several interrupts are used to manage different aspects of the robot's operation, including timer interrupts for timing tasks, ADC interrupts for sensor readings, and external interrupts for user input.

Autonomous Behavior:

- The robot features autonomous behavior, reacting to sensor inputs to navigate its environment. It adjusts its movement based on sensor readings, indicating capabilities for line following or obstacle avoidance. It also communicates movement back to the users device.

Overall Performance:

- The project demonstrates effective utilization of the microcontroller's peripherals to achieve the desired functionality. The code structure is organized, with clear separation of tasks and efficient use of hardware resources.

Areas for Improvement:

- While the code appears functional, there may be opportunities for optimization or refinement, such as improving sensor calibration or enhancing the autonomous navigation algorithm.

Future Development:

- Possible future enhancements could include implementing the optional content, adding more sensors for environment perception, and or integrating wireless communication for remote control or data logging purposes.

In summary, the project showcases the capabilities of the microcontroller in controlling an autonomous robot and lays the foundation for further exploration and development in robotics and embedded systems.