Mini-Project

Report

|  |  |
| --- | --- |
| **Course code and name:** | B31DD – Embedded Systems |
| **Type of assessment:** | **Individual** |
| **Coursework Title:** | Mini-Project |
| **Student Name:** | Atique Arif Kondvilkar |
| **Student ID Number:** | H00431858 |

|  |
| --- |
| **Declaration of authorship. By signing this form:**   * **I declare** that the work I have submitted for individual assessment OR the work I have contributed to a group assessment, is entirely my own. I have NOT taken the ideas, writings or inventions of another person and used these as if they were my own. My submission or my contribution to a group submission is expressed in my own words. Any uses made within this work of the ideas, writings or inventions of others, or of any existing sources of information (books, journals, websites, etc.) are properly acknowledged and listed in the references and/or acknowledgements section. * I confirm that I have read, understood and followed the University’s Regulations on plagiarism as published on the [University’s website](https://www.hw.ac.uk/uk/students/studies/examinations/plagiarism.htm), and that I am aware of the penalties that I will face should I not adhere to the University Regulations. * I confirm that I have read, understood and avoided the different types of plagiarism explained in the University guidance on [Academic Integrity and Plagiarism](https://heriotwatt.sharepoint.com/sites/skillshub/SitePages/Academic-Integrity-and-Plagiarism.aspx)   **Student Signature** *(type your name): Atique Kondvilkar*  **Date**: *30/11/2023* |

# Introduction

The selected mini project is about creating a mobile robot with the Arduino UNO microcontroller, which is well-known, is in charge of all of its essential functions. The brains behind this project are the Arduino UNO, which is renowned for being affordable, accessible, and open source. The main goal of the project is to build a mobile buggy robot that can be controlled by Bluetooth. To do this, an HC-05 Bluetooth module is used to create a smooth connection between the Arduino UNO and a smartphone. Through serial communication made possible by this link, the Arduino can process orders from the smartphone and adjust the robot's motor directions to impart a variety of functions.

With the use of six different approaches, this little project provides an in-depth investigation of embedded systems. These methods include servo control implementation, timer use, input-output handling, interrupt management, serial connection integration, and LCD display incorporation. To improve the robot's skills and performance, each of these methods is essential.

The utilization of the C programming language is essential for carrying out various activities in the project, and Atmel Studio is the preferred development environment. The project effectively integrates hardware and software components through the use of Atmel Studio, making writing, debugging, and deployment onto the Arduino UNO quick and easy.

Essentially, this small project offers insights into the complexities of embedded systems and is a priceless learning tool. It not only emphasizes the usefulness of different approaches in real-world situations but also highlights the Arduino UNO microcontroller's adaptability. The project embodies an integrated approach to comprehending and utilizing embedded systems in practical applications by integrating several features, such as motor control and serial communication.

# Functionalities

The wheels of this Arduino UNO-controlled mobile robot project are driven by DC motors, which allow for smooth motion. The Arduino UNO interprets commands using input-output functionalities and a Bluetooth module for serial connection, allowing for precise control over the robot's motions. The introduction of interrupts provides a way to rank certain operations in order of importance, and effective work completion. Moreover, servo motor control increases the robot's range of motion. Real-time feedback is provided via the incorporation of an LCD, which visibly displays the actions of the robot. This is a complex project that combines several different aspects, such as interrupt management and motor control, to create a cohesive whole for embedded systems on a small, portable platform.

# Arduino schematics diagram

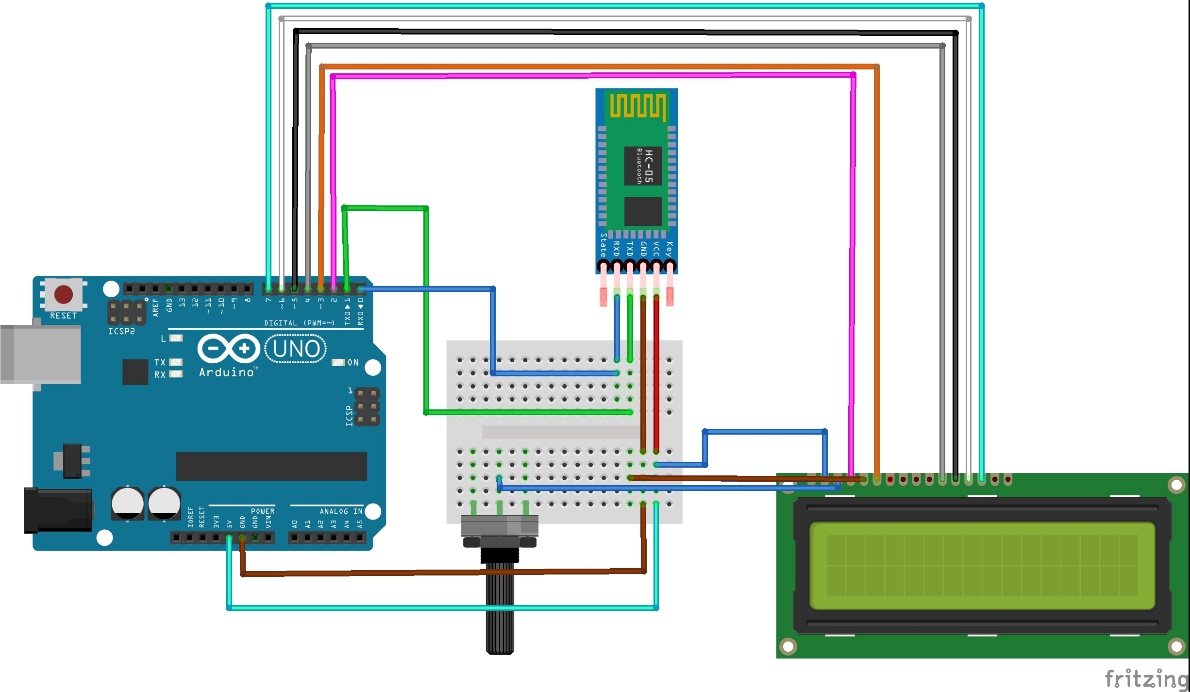
## Schematics diagram 1

A circuit board with wires

Description automatically generated

DC Motors, Servo Motor and LED Interrupt

## Schematics diagram 2



HC-05 and OLED Display

# Explanation of techniques

Timer:

Timer us used to add a delay function to delay the blinking of the light while the interrupt happens. The delay function will call when the light operates between high to low.

Interrupt:

The interrupt technique will occur when the button is pressed, as the button is pressed the interrupt function will get involved to interrupt all the ongoing processes and do the assigned function at higher priority.

Servo Control:

The Servo is added to do the specific task as a function is added when it will be called by the case. The Servo must be implemented in different directions as defined.

Input/output:

This technique is used to define the peripherals of the components as input/output such as buttons, sensors, communication devices, actuators etc.

Serial Communication:

Serial Communication is used to communicate between the sensors as input and a microcontroller to transmit data as assigned.

LCD Display:

The display is used to display the given text as pixels on the LCD screen. As I have used it to print the name of the robot and it was made by.

# Code in C Language

1. For Bluetooth Car

#include <avr/io.h>

#include <util/delay.h>

#include <avr/interrupt.h>

#define *F\_CPU* 16000000UL // Assuming a 16MHz clock frequency

#define BAUDRATE 9600

#define BAUD\_PRESCALLER (((*F\_CPU* / (BAUDRATE \* 16UL))) - 1)

void delay\_500ms(); //Function containing the delay of 500ms

ISR (INT0\_vect); //Function to be executed when Interrupt happens

void USART\_init(void) {

UBRR0H = (*uint8\_t*)(BAUD\_PRESCALLER >> 8); // Set baud rate

UBRR0L = (*uint8\_t*)(BAUD\_PRESCALLER);

UCSR0B = (1 << RXEN0) | (1 << TXEN0); // Enable transmit

UCSR0C = (3 << UCSZ00); // Set 8-bit (default)

}

unsigned char USART\_receive(void) {

while (!(UCSR0A & (1 << RXC0))); // Wait to receive data

return UDR0; // Read data from UDR

}

void USART\_transmit(unsigned char data) {

while (!(UCSR0A & (1 << UDRE0))); // Wait for empty transmit buffer

UDR0 = data; // Put data into buffer, sends the data

}

void USART\_putstring(const char\* string) {

while (\*string != 0x00) {

USART\_transmit(\*string);

string++;

}

}

void initMotors() {

// Set motor control pins as outputs

DDRD |= (1 << PD3) | (1 << PD4);

DDRB |= (1 << PB3) | (1 << PB4);

}

void forward() {

PORTD |= (1 << PD3); // Set left motor pin for forward motion

PORTD &= ~(1 << PD4); // Clear opposite left motor pin

PORTB |= (1 << PB4); // Set right motor pin for forward motion

PORTB &= ~(1 << PB3); // Clear opposite right motor pin

}

void backward() {

PORTD |= (1 << PD4); // Set left motor pin for backward motion

PORTD &= ~(1 << PD3); // Clear opposite left motor pin

PORTB |= (1 << PB3); // Set right motor pin for backward motion

PORTB &= ~(1 << PB4); // Clear opposite right motor pin

}

void left() {

PORTD &= ~(1 << PD3 | 1 << PD4); // Clear both left motor pins

PORTB |= (1 << PB4); // Set right motor pin for turning left

PORTB &= ~(1 << PB3); // Clear opposite right motor pin

}

void right() {

PORTB &= ~(1 << PB3 | 1 << PB4); // Clear both right motor pins

PORTD |= (1 << PD3); // Set left motor pin for turning right

PORTD &= ~(1 << PD4); // Clear opposite left motor pin

}

void stopMotors() {

PORTD &= ~(1 << PD3 | 1 << PD4); // Clear both left motor pins

PORTB &= ~(1 << PB3 | 1 << PB4); // Clear both right motor pins

}

void delay\_500ms(){

TCNT1=0; //Initializing Timer 1 in Normal mode

TCNT1H = 0x1E; //Loading Time High as 1E

TCNT1L = 0x84; //Loading Timer Low as 84 Together equivalent to decimal value 35938

TCCR1A = 0; //Running Timer in normal mode

TCCR1B = 0x05; //Running in 1024 prescaler

while((TIFR1&(1<<OCF1A))==0){} //wait until OCF1A is set

TCCR1B = 0; //Stop Timer 1

TIFR1 = 1<<OCF1A; //Clear the flag

}

ISR (INT0\_vect)

{

*uint8\_t* switchState = PIND & ( 1 << PD2 ); //variable to save the switch state

if (switchState) //checking for switch press

{

PORTB ^= (1<<5); //Toggle LED1

delay\_500ms(); //delay 500ms

PORTB ^= (1<<5);

delay\_500ms();

}

}

void servoMotor(){

DDRB |= 1 << PINB1; //Initialise Pin 9 as Output

/\* 1. Set Fast PWM mode 14: WGM11, WGM12, WGM13 to 1\*/

/\* 2. Set pre-scale of 8 \*/

/\* 3. Set Fast PWM non-inverting mode \*/

TCCR1A |= (1 << WGM11) | (1 << COM1A1);

TCCR1B |= (1 << WGM13) | (1 << WGM12) |(1 << CS11);

int delay = 10000; //Short Delay between angles

/\* 4. Set ICR1: ICR1 is the top defining PWM period \*/

ICR1 = 40000;

while(1) {

OCR1A = 1000; //-90

*\_delay\_ms*(delay);

OCR1A = 2000; //-45

*\_delay\_ms*(delay);

OCR1A = 3000; //0

*\_delay\_ms*(delay);

OCR1A = 4000; //45

*\_delay\_ms*(delay);

OCR1A = 5000; //90

*\_delay\_ms*(delay);

OCR1A = 4000; //45

*\_delay\_ms*(delay);

OCR1A = 3000; //0

*\_delay\_ms*(delay);

OCR1A = 2000; //-45

*\_delay\_ms*(delay);

}

}

void ledInterrupt(){

DDRB |= 1<<5; //Initializing PB 5 as output i.e Pin 13 in Arduino

EICRA |= (1 << ISC00); // set INT0 to trigger on ANY logic change

EIMSK |= (1 << INT0); // Turns on INT0, Declaring the interrupt on INT0 i.e digital pin 2 in Arduino

sei(); // turn on interrupts

}

int main() {

USART\_init(); // USART initialization

initMotors();

while (1) {

char ReceivedChar = USART\_receive(); // Wait until data is received

switch (ReceivedChar) {

case 'S':

stopMotors();

USART\_putstring("Robot stopped!\n");

break;

case 'F':

forward();

USART\_putstring("Robot moving forward!\n");

break;

case 'B':

backward();

USART\_putstring("Robot moving backward!\n");

break;

case 'L':

left();

USART\_putstring("Robot turning left!\n");

break;

case 'R':

right();

USART\_putstring("Robot turning right!\n");

break;

case 'V':

servoMotor();

break;

case 'W':

ledInterrupt();

break;

}

}

}

1. For LCD Display and Bluetooth Module

#include <avr/io.h>

#include <avr/io.h>

#include <util/delay.h>

#define LCD\_RS PD0

#define LCD\_E PD1

#define LCD\_D4 PD4

#define LCD\_D5 PD5

#define LCD\_D6 PD6

#define LCD\_D7 PD7

void LCD\_Command(unsigned char cmnd) {

// Sending upper nibble

PORTD = (PORTD & 0x0F) | (cmnd & 0xF0);

PORTD &= ~ (1<<LCD\_RS); // RS = 0 for command

PORTD |= (1<<LCD\_E); // Enable pulse

*\_delay\_us*(1);

PORTD &= ~ (1<<LCD\_E);

*\_delay\_us*(200);

// Sending lower nibble

PORTD = (PORTD & 0x0F) | (cmnd << 4);

PORTD |= (1<<LCD\_E);

*\_delay\_us*(1);

PORTD &= ~ (1<<LCD\_E);

*\_delay\_ms*(2);

}

void LCD\_Char(unsigned char data) {

// Sending upper nibble

PORTD = (PORTD & 0x0F) | (data & 0xF0);

PORTD |= (1<<LCD\_RS); // RS = 1 for data

PORTD |= (1<<LCD\_E);

*\_delay\_us*(1);

PORTD &= ~ (1<<LCD\_E);

*\_delay\_us*(200);

// Sending lower nibble

PORTD = (PORTD & 0x0F) | (data << 4);

PORTD |= (1<<LCD\_E);

*\_delay\_us*(1);

PORTD &= ~ (1<<LCD\_E);

*\_delay\_ms*(2);

}

void LCD\_Init() {

DDRD = 0xFF; // Make PORTD as output

*\_delay\_ms*(20); // LCD Power ON delay

LCD\_Command(0x02); // 4-bit mode

LCD\_Command(0x28); // Initialize in 4-bit, 2 lines, 5x7 mode

LCD\_Command(0x0c); // Display ON, Cursor OFF

LCD\_Command(0x06); // Auto Increment cursor

LCD\_Command(0x01); // Clear display

*\_delay\_ms*(2);

}

void LCD\_String(char \*str) {

int i;

for(i=0; str[i]!=0; i++) {

LCD\_Char(str[i]);

}

}

void LCD\_Clear() {

LCD\_Command(0x01); // Clear display

*\_delay\_ms*(2);

}

int main() {

LCD\_Init(); // Initialization of LCD

*\_delay\_ms*(200);

LCD\_String("Mobile Robot"); // Write string on 1st line of LCD

LCD\_Command(0xC0); // Go to 2nd line

LCD\_String("BY: Atique");

while(1);

}

# Component List

|  |  |
| --- | --- |
| Components | Quantity |
| 1. At-mega 328p Microcontroller | 1 |
| 1. L289n motor driver | 1 |
| 1. HC-05 Bluetooth Module | 1 |
| 1. DC Motor | 4 |
| 1. Servo Motor | 1 |
| 1. Battery 9V | 1 |
| 1. Battery Connector | 1 |
| 1. LCD Display | 1 |
| 1. Car Chaise | 1 |
| 1. Car Wheel | 4 |
| 1. Bread Board | 1 |
| 1. Button | 1 |
| 1. LED Display | 1 |
| 1. Resistor | 2 |

# Results

I've successfully developed a mobile robot for a mini project that uses wheels powered by DC motors and Bluetooth serial connectivity to enable remote control. I was able to train the robot to perform tasks in response to different inputs by interacting with it via a mobile remote. Every order from the remote causes the robot to perform a preprogrammed action, like turning, stopping, or going forward. The robot's movements may be controlled with ease and responsiveness thanks to this configuration. Writing the software to translate the remote inputs into tasks the robot could perform, building a dependable Bluetooth communication protocol, and integrating motion-related hardware components were all part of the development process. After these features were successfully implemented, a live demonstration was given to show off the robot's capacity to carry out its assigned responsibilities.