# Faculty of Information Technology

# Automated Camera Stand

Group 47


| | |
|---|---|
| **P.A.U.D. Herath** | **204074M** |
| P.H.P. Jayathilaka | 204087F |
| D.M.B.M. Dissanayake | 204047J |
| S.P.S.N. Pathirana | 204150T |
| A.M.D.B. Rathnayaka | 204179N |



Supervisor's Name: Mr. B.H. Sudantha
Dean/Senior Lecturer
Faculty of Information Technology

Ms. Chanduni Gamage
Lecturer
Dept. of Information Technology


Signature of Supervisor: ……. …….

Date of Submission: 24/09/2021

## Table of Content

## Table of Figures

## Introduction

Wildlife photography is a popular category of photography, done by beginners, enthusiasts, and professionals. When taken technically, it involves capturing any type of animal, from birds to insects to butterflies to mammals. But wildlife photographers most commonly take photos of mammals, reptiles, amphibians, and birds. Wildlife photography is a loosely defined profession that demands a passion for nature and art. These photographers make a career of traveling to remote areas and taking pictures of wild animals and natural scenery with a risk.[01]

Wildlife photographers are some of the world's most valued professionals. According to the U.S. Bureau of Labor Statistics, the average annual wage of most wildlife photographers was $50,290 per year, or $24.18 per hour, as of May 2020. So, this is a higher-paying job. Due to this reason, many new photographers have come to this field. And most wildlife photographers are freelancers. The amount of money that a freelance wildlife photographer makes is largely determined by his talent and ability to get decent-paying work. From all these things it is crystal clear that wildlife photography is one of the best careers in the world.

Wildlife photography is one of the most dangerous professions in the world. The following instances are examples of the dangerous which happen for wildlife photographers. One such incident happened in May of 2000 when a female wildlife photographer was attacked and partially eaten by a 112-pound female black bear in Tennessee. Last year in Colorado a wildlife biologist and photographer, Tom Mussel, got too close to a cow elk and her calf, and he was attacked when he stumbled as he tried to escape the charging cow. Elk and deer will attack humans when they feel cornered or threatened. A southern California man killed by a grizzly bear in Alaska's backcountry was shooting photos of the animal that killed him just moments before the attack, a National Park Service official said Sunday. The bear that killed Richard White, 49, was still with his body when rangers found him in Danail National Park, the official said. Photographs found in his camera revealed that White was watching the bear for at least eight minutes near a river before the attack [02].

From our project, we mainly focused on avoiding the danger for the wildlife photographer and the safety of the camera. In this period many wildlife photographers are dying while taking photos due to animal attacks. We are going to introduce a new device to avoid such a problem. It is an Automated Camera Stand.

## Literature Survey

In modern cameras, we have the feature to take remote photos. We call it remote photography. Therefore, we can take pictures even if we are far from the camera. But there is no way to bring the camera near to the animals rather than a person carrying the camera.

From the photographer's side:

- Danger to the life of the photographer.
- Time wastage of the photographer.
- Can't find a good angle to take a photo.
- Can't get close enough to the animal.
- Constantly changing lighting conditions.
- No safety to the camera.

From the animal's side:

- Animals get scared when we try to reach them.
- The behaviors of the animals may change.

From the environment side:

- Bad impact on biodiversity.

Therefore, when we try to find out a solution for this we came across with some similar projects.

**Beetle cam for wildlife photography**

*Figure 1 Beetle cam*

Beetle cam has designed to take wildlife photographs. Wildlife photographer Will Burrard-Lucas had long wanted to add up-close-and-personal images of iconic African animals to his portfolio. But to get those intimate shot of lions and leopards, he would need to crawl up right next to their sharp-toothed faces [03]. This beetle cam can move forward and backward by using the remote controller. And also, beetle cam can be turned by using the wheels.

In our product we have all the functionalities same as the beetle cam. But there are some uncommon functionalities rather than this beetle cam. There is a special feature that can move the camera to the location easily by using GPS location guiding and can avoid obstacles itself. And also, can rotate the camera vertically using the remote controller. There is a siren for the safety of the camera.

## Aim & Objectives

### Aim

This project aims to the distance photography process to make sure it is safer and effective.

### Objectives

The objectives of the project are as follows,

- Rotate the camera both vertically and horizontally.
- Move the stand according to the given GPS coordinates.
- Reach the target safely.
- Protect the camera from the animals by using the alarm.
- Control the camera holder using a remote controller.

## Analysis and Design

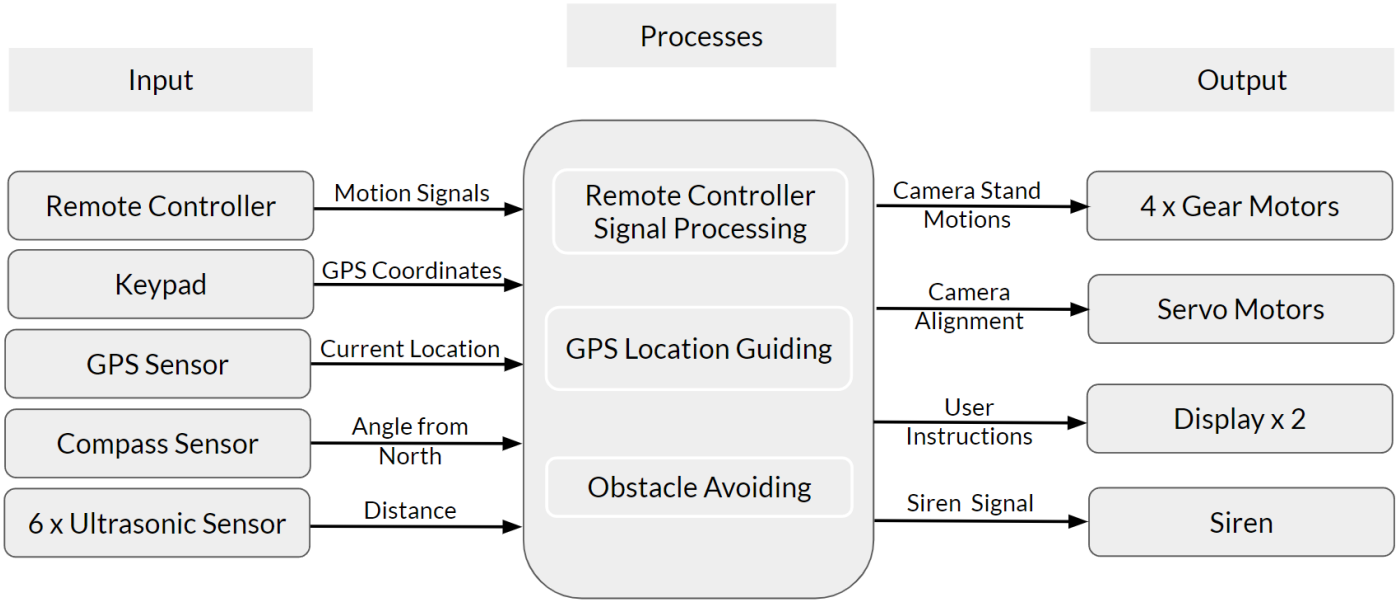



*Figure 2 Block Diagram of Proposed System*

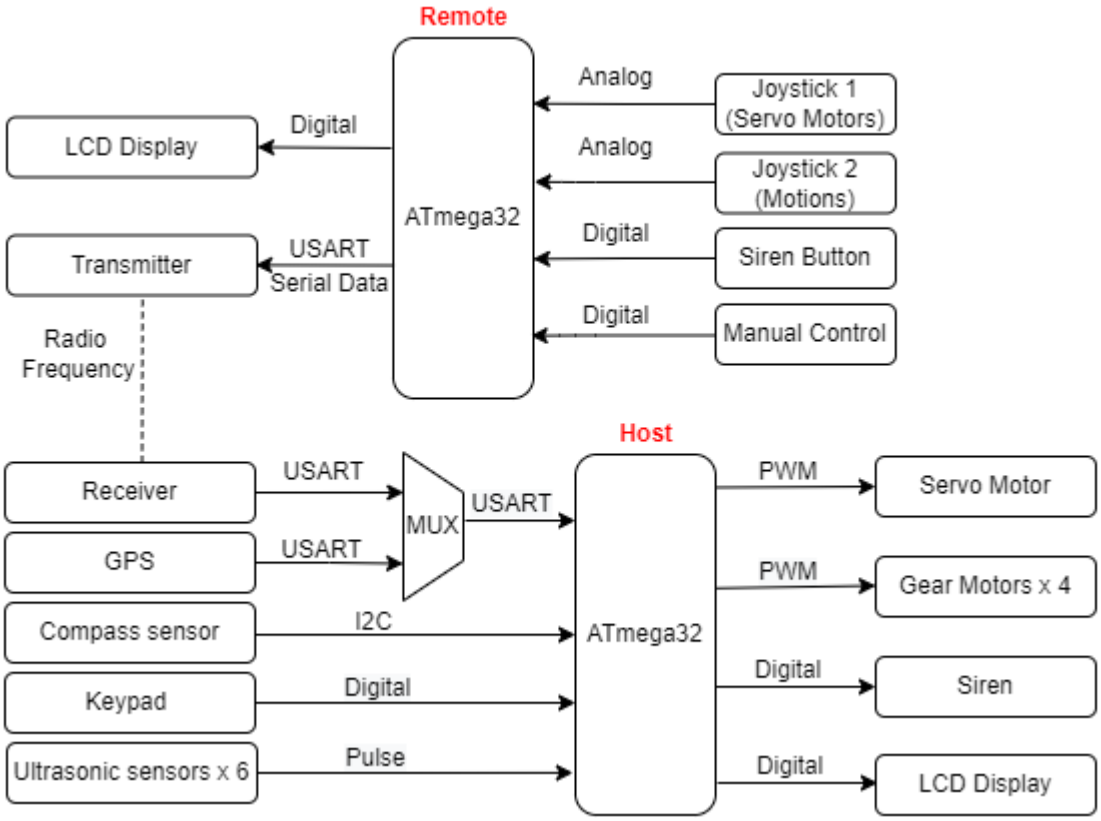



*Figure 3 System Diagram of Proposed System*

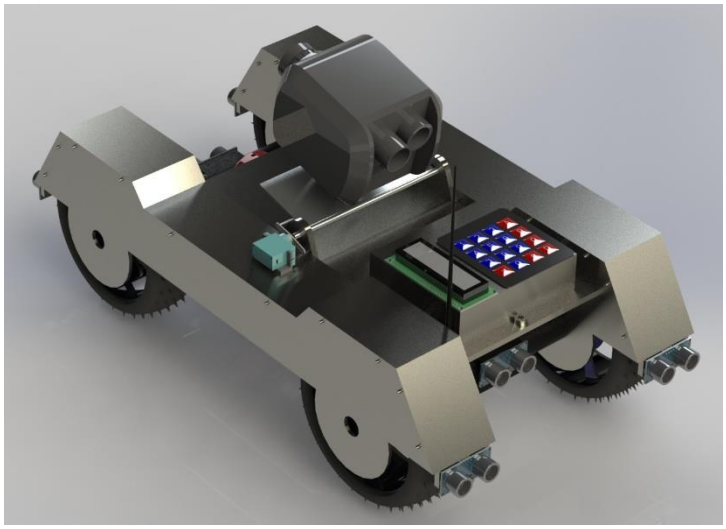


*Figure 6 Back upper 3D view*

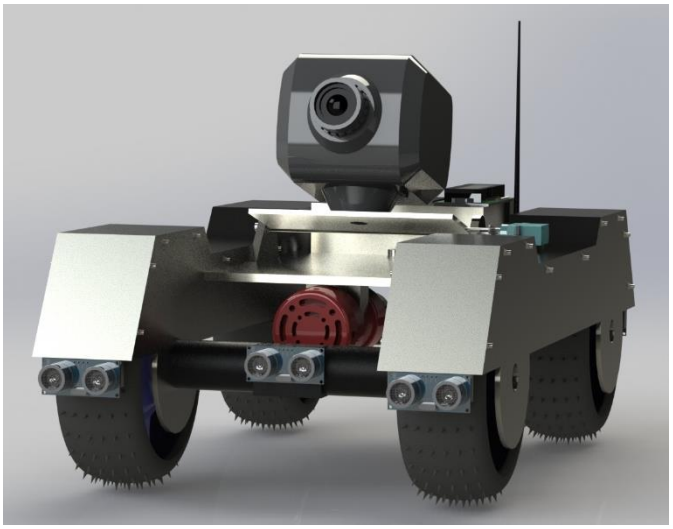


*Figure 7 Front lower 3D view*

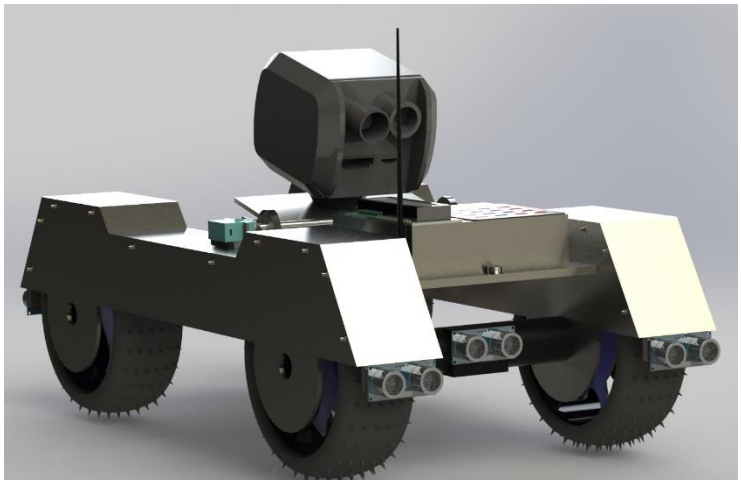


*Figure 4 Back lower 3D view*

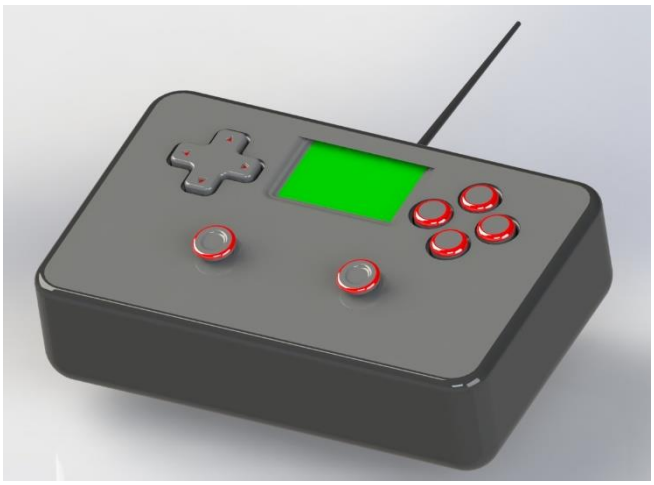


*Figure 5 Remote controller 3D view*

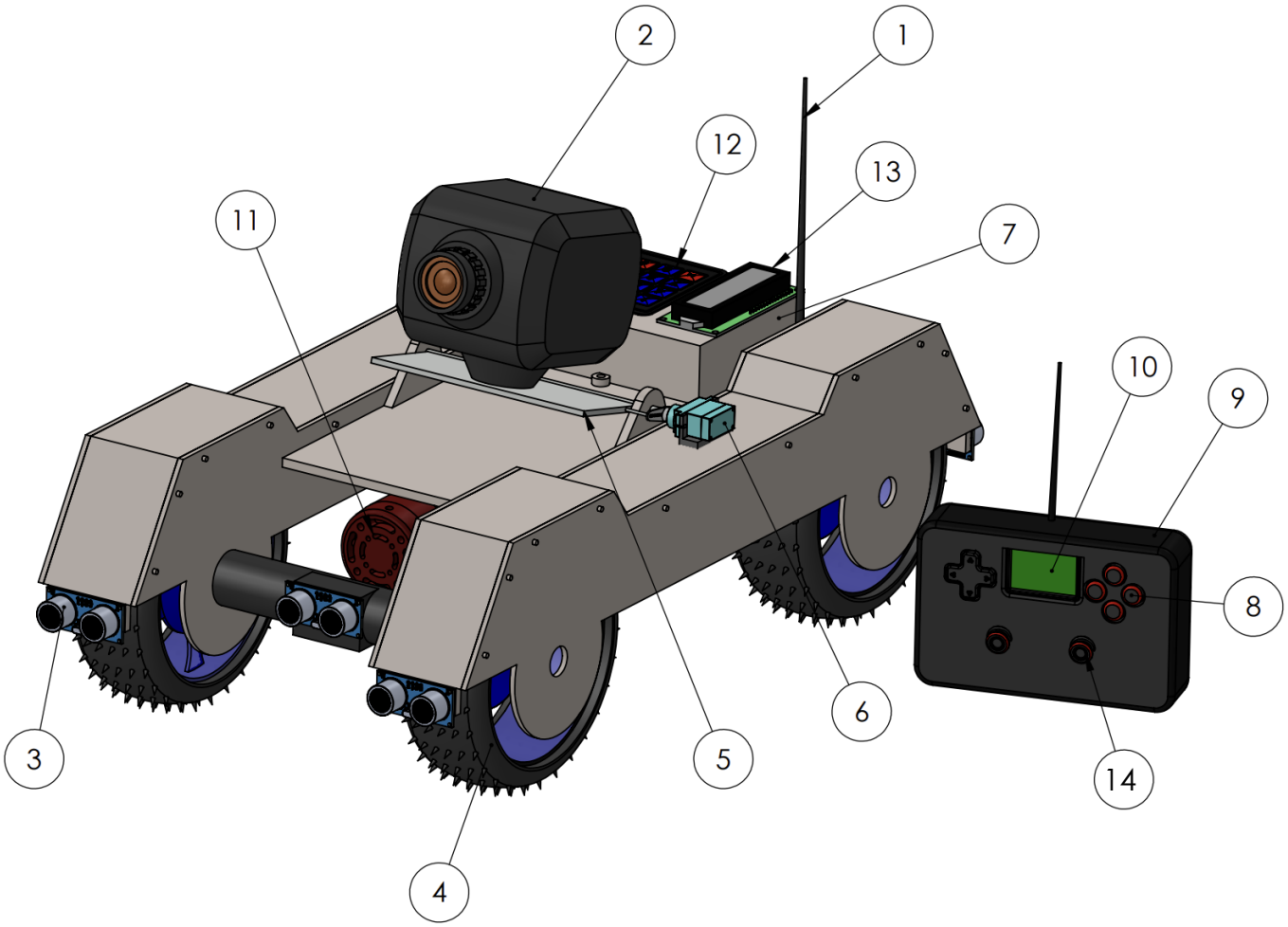



*Figure 8 Named Camera Stand 3D view*

1. RF Antenna
2. Camera
3. Ultrasonic sensors
4. Wheels with gear motors
5. Universal camera mount
6. Servo motor
7. Project box
    a. Atmega32 IC
    b. GPS module
    c. Compass sensor
    d. RF receiver
8. Remote controller buttons
9. Remote controller
10. LCD Display
11. Siren
12. 4x4 Numpad
13. LCD Display
14. Joystick

## Testing and Implementation

In our project, there are two ATmega32 microcontrollers, one is for remote controller and the other one is for the host (Camera stand). The microcontroller takes inputs related to motions and angles from two joysticks. The microcontroller of the remote processes the data and transmit from remote to host through RF (Radio Frequency) modules.

Then the microcontroller of the host gets data from the RF receiver module and GPS module using a 2×1 multiplexer. And also, the microcontroller of the host takes inputs from ultrasonic sensors, compass sensor and the keypad. The microcontroller of the host processes that data and send data to the servo motors and gear motors. And also send digital signals to the display and the siren.

### ATmega32 Microcontroller


*Figure 10 Atmega32 Physical view*

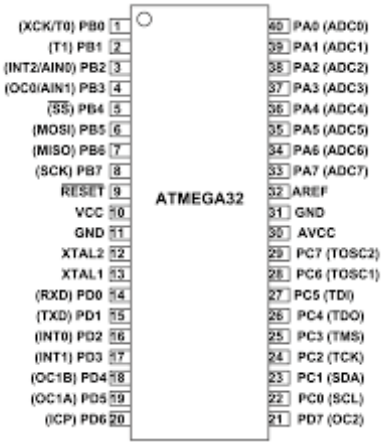



*Figure 9 Atmega32 Pin Configuration*

### Thumb Joystick

These joystick modules are used to control movements of the camera stand and the vertical camera angle.


*Figure 12 Thumb Joystick*

```
/*
 * Analog input (0 - 255)
 *
 * Parameter
 *  - pin (string) - Input pin eg: A1, B4
 * Return
 *  - (int) - Value of the pin 0 or 255.
 */
int ADC_read(Pin pin) {
    // Change Configuration
    ADMUX = (1 << ADLAR); // ADC Left Adjust Result; Set to Left-Justified

    if (pin.port != 'A') { // A port is the only port ADC accessible.
        return 0;
    }

    ADMUX &= 0xE0; // Clean multiplexer bits
    ADMUX |= pin.pin; // Set input channel

    ADCSRA |= (1 << ADSC); // ADC Start Conversion

    uint8_t adif = (1 << ADIF); // Get value of ADC Interrupt Flag
    while((ADCSRA & adif) == 0); // Wait until ADC Interrupt Flag to be 1

    int val = ADCH; // Read value

    // Change Configuration back to previous
    ADMUX = (0 << ADLAR); // ADC Left Adjust Result; Set to Right-Justified
```

*Figure 11 Read ADC value code sample*

**Liquid Crystal 16 × 2 Display**

*Figure 13 LCD Display Physical view*

These LCD displays are used to show GPS coordinates and the current remote's configurations. By using this we can convert camera stand more user friendly.

```
Pin Configuration.
+---------+-------------+
| Display |    Atmel    |
|---------+-------------|
|VSS      | GND         |
|VCC      | 5v          |
|VEE      | 10K Preset  |
|RS       | B0          |
|RW       | GND         |
|En       | B1          |
|D4       | B4          |
|D5       | B5          |
|D6       | B6          |
|D7       | B7          |
+---------+-------------+

*/

#define LCD_Dir  DDRB // Define LCD data port direction
#define LCD_Port PORTB // Define LCD data port
#define RS PB0 // Define Register Select pin
#define EN PB1 // Define Enable signal pin

void LCD_Command(unsigned char cmnd) {
    LCD_Port = (LCD_Port & 0x0F) | (cmnd & 0xF0); // Sending upper nibble
    LCD_Port &= ~ (1 << RS); // RS=0, command reg.
    LCD_Port |= (1 << EN);  // Enable pulse
    _delay_us(1);
    LCD_Port &= ~ (1 << EN);

    _delay_us(200);

    LCD_Port = (LCD_Port & 0x0F) | (cmnd << 4); // Sending lower nibble
    LCD_Port |= (1 << EN);
    _delay_us(1);
    LCD_Port &= ~(1 << EN);
    _delay_ms(2);
}


void LCD_Char(unsigned char data) {
    LCD_Port = (LCD_Port & 0x0F) | (data & 0xF0); // sending upper nibble
```

*Figure 14 Code for LCD Display*

### Servo motor

This servo motor is used to change the vertical angle of the camera.


*Figure 15 Servo motor Physical view*

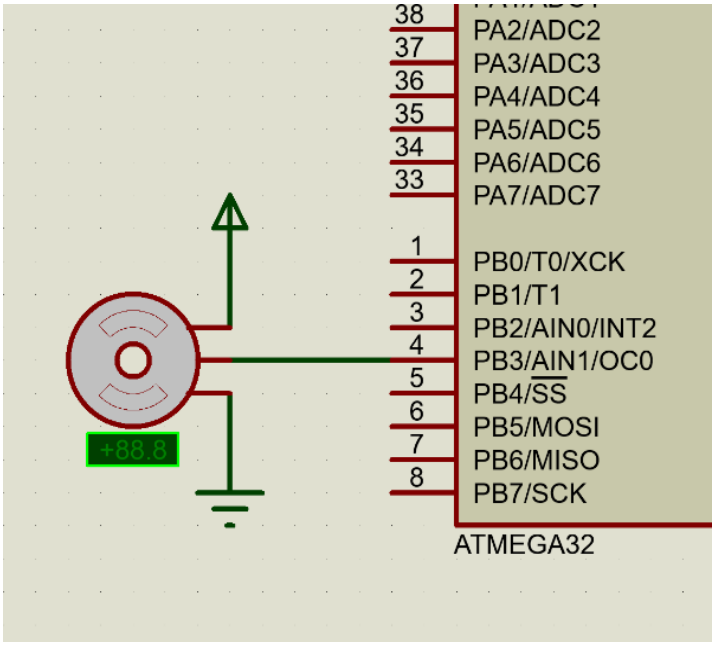



*Figure 16 Servo motors proteus schematic view*

### Moter drive

Motor drive is used to amplify the PWM signal from the Atmega32 and supply it to the gear motors.

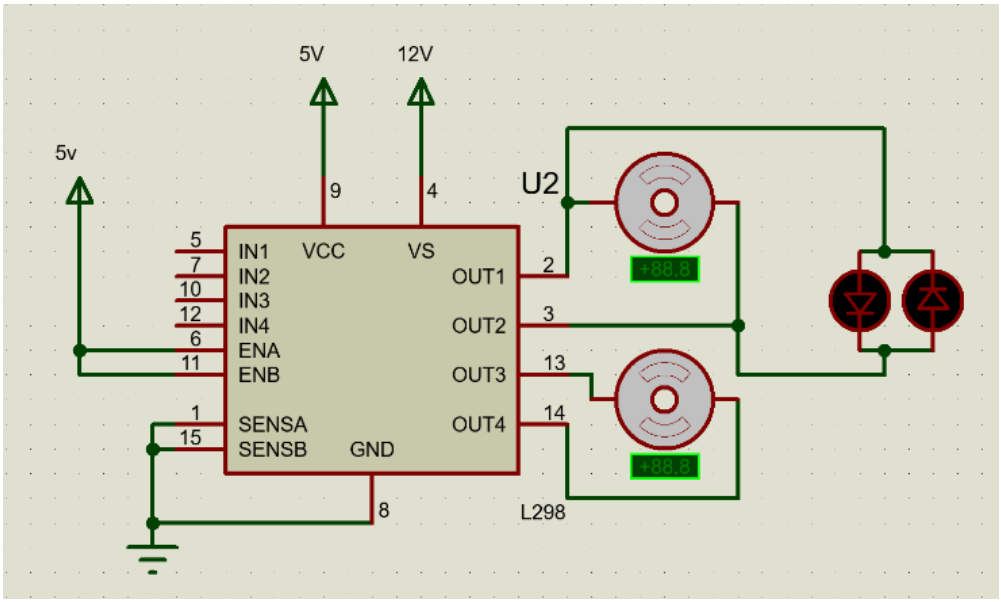



*Figure 17 L298N IC Config for motors*

## Screen shots of the project.

```c
#include "../defines.h"


/*
 * Digital output
 *
 * Parameter
 *  - pin (string) - Input pin eg: A1, B4
 *  - level (int) - 1 for high value, 0 for low value
 * Return
 *  - (int) - 0 if no errors.
 */
int digital_write(Pin pin, int level) {
    volatile uint8_t *regi = select_register(pin.port, &PORTA, &PORTB, &PORTC, &PORTD); // Select PORT register according to pin

    if (level == 1) { // Check weather high or low
        *regi |= 1 << pin.pin; // Output high value
    } else {
        *regi &= ~(1 << pin.pin); // Output low value
    }
    return 0;
}


/*
 * Digital input
 *
 * Parameter
 *  - pin (string) - Input pin eg: A1, B4
 * Return
 *  - (int) - Value of the pin 0 or 1.
 */
int digital_read(Pin pin) {
    volatile uint8_t *regi = select_register(pin.port, &PINA, &PINB, &PINC, &PIND); // Select PIN register according to pin

    int val = *regi & (1 << pin.pin); // Get value of the register
    if (val == 0) { // Check weather the value is high or low
        return 0; // Return low value
    } else {
        return 1; // Return high value
    }
}
```

*Figure 19 Digital input and output code*

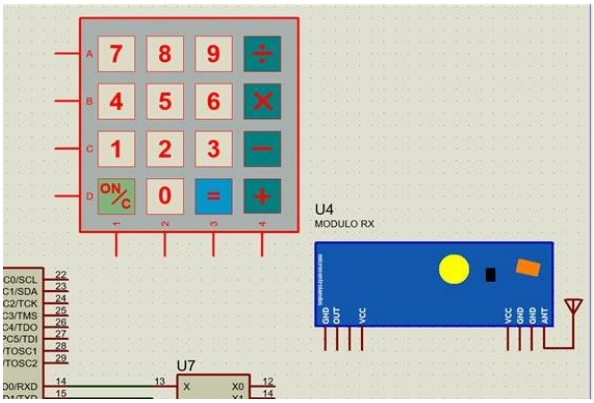



*Figure 18 4x4 Keypad and RF receiver Proteus view*

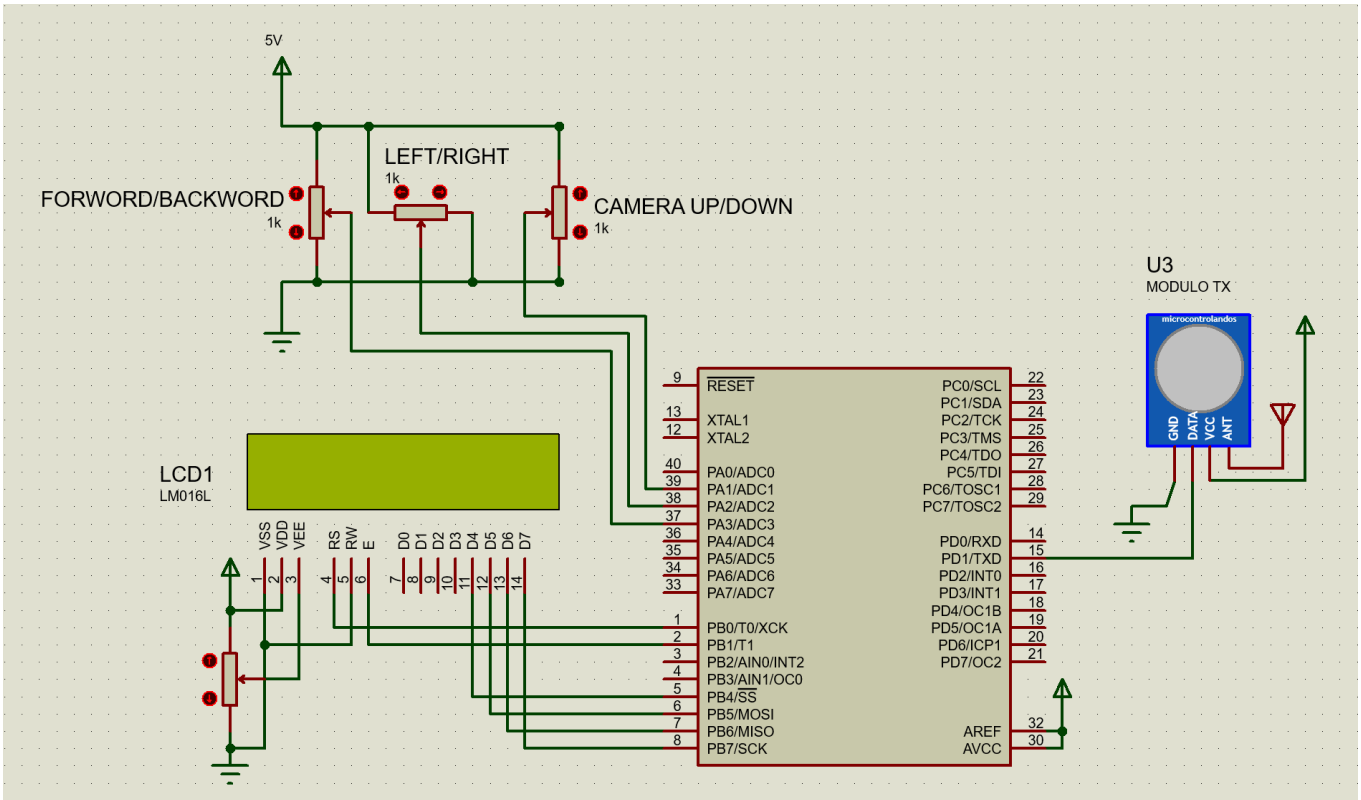



*Figure 20 Remote controller circuit Proteus schematic view*

# Previous Action Plan

| Action Plan | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| # | Responsible group member | Task | Duration | Start Date | End Date | Completion Date | | | | | | | |
| | | | | | | 20/09/2021 | 27/09/2021 | 4/10/2021 | 11/10/2021 | 18/10/2021 | 25/10/2021 | 1/11/2021 | 8/11/2021 |
| 1 | 204074M | GPS sensor | 2 week | 20/09/2021 | 3/10/2021 | | | | | | | | |
| | | Compass sensor | 2 week | 4/10/2021 | 17/10/2021 | | | | | | | | |
| | | GPS location guiding | 3 week | 01/11/2021 | 21/11/2021 | | | | | | | | |
| | | | | | | | | | | | | | |
| 2 | 204087F | Ultrasonic sensors | 3 week | 20/09/2021 | 10/10/2021 | | | | | | | | |
| | | Gear motors with wheels | 1 week | 11/10/2021 | 17/10/2021 | | | | | | | | |
| | | Obstacle Avoiding | 2 week | 18/10/2021 | 31/10/2021 | | | | | | | | |
| | | Motor Controlling | 3 week | 01/11/2021 | 21/11/2021 | | | | | | | | |
| | | LCD display | 1 week | 18/10/2021 | 24/10/2021 | | | | | | | | |
| | | Keypad | 1 week | 25/10/2021 | 31/10/2021 | | | | | | | | |
| | | | | | | | | | | | | | |
| 3 | 204047J | First Joysticks - For servo motors | 3 week | 20/09/2021 | 10/10/2021 | | | | | | | | |
| | | RF Transmitter | 3 week | 11/10/2021 | 31/10/2021 | | | | | | | | |
| | | Programming | 3 week | 1/11/2021 | 21/11/2021 | | | | | | | | |
| | | | | | | | | | | | | | |
| 4 | 204150T | Servo motors | 2 week | 20/09/2021 | 03/10/2021 | | | | | | | | |
| | | Alarm | 2 week | 04/10/2021 | 17/10/2021 | | | | | | | | |
| | | Triger Alarm | 2 week | 18/10/2021 | 31/10/2021 | | | | | | | | |
| | | Servo Controlling | 3 week | 01/11/2021 | 21/11/2021 | | | | | | | | |
| | | | | | | | | | | | | | |
| 5 | 204179N | Second Joystick - For motions | 3 week | 20/09/2021 | 10/10/2021 | | | | | | | | |
| | | RF Receiver | 3 week | 11/10/2021 | 31/10/2021 | | | | | | | | |
| | | Programming | 3 week | 1/11/2021 | 21/11/2021 | | | | | | | | |

# Action Plan for Remaining Work

| Action Plan | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| # | Responsible group member | Task | Duration | Start Date | End Date | Completion Date | | | | | |
| | | | | | | 31/12/2021 | 31/1/2022 | 1/2/2022 | 1/3/2022 | 1/4/2022 | 9/5/2022 |
| 1 | 204074M | Compass sensor | 1 month | 31/12/2021 | 31/1/2022 | | | | | | |
| | | GPS location guiding | 1 month | 1/2/2022 | 1/3/2022 | | | | | | |
| | | Test and implementation | 2 month | 2/3/2022 | 9/5/2022 | | | | | | |
| | | | | | | | | | | | |
| 2 | 204087F | Obstacle Avoiding | 1 month | 31/12/2021 | 31/1/2022 | | | | | | |
| | | Motor Controlling | 1 month | 1/2/2022 | 1/3/2022 | | | | | | |
| | | Test and implementation | 2 month | 2/3/2022 | 9/5/2022 | | | | | | |
| | | | | | | | | | | | |
| 3 | 204047J | First Joysticks - For servo motors | 1 month | 31/12/2021 | 31/1/2022 | | | | | | |
| | | RF Transmeter | 1 month | 1/2/2022 | 1/3/2022 | | | | | | |
| | | Test and implementation | 2 month | 2/3/2022 | 9/5/2022 | | | | | | |
| | | | | | | | | | | | |
| 4 | 204150T | Servo motors | 1 month | 31/12/2021 | 31/1/2022 | | | | | | |
| | | Test and implementation | 1 month | 1/2/2022 | 1/3/2022 | | | | | | |
| | | Trigger Alarm | 1 month | 2/3/2022 | 1/4/2022 | | | | | | |
| | | Servo Controlling | 1 month | 2/4/2022 | 9/5/2022 | | | | | | |
| | | | | | | | | | | | |
| 5 | 204179N | Second Joystick - For motions | 1 month | 31/12/2021 | 31/1/2022 | | | | | | |
| | | RF Receiver | 1 month | 1/2/2022 | 1/3/2022 | | | | | | |
| | | Test and implementation | 2 month | 2/3/2022 | 9/5/2022 | | | | | | |

## References


[1] A. Duke, “Hiker photographs bear just before fatal grizzly attack,” *CNN*, 26-Aug-2012.

[2] J. Dempsey, “Introduction to wildlife photography: A guide for beginners,” *Phototraces.com*, 03-Jan-2020. [Online]. Available: https://www.phototraces.com/b/wildlife-photography/. [Accessed: 28-Nov-2021].

[3] M. Gallucci, “These robots are transforming how we see wildlife,” *Mashable*, 12-Jan-2017. [Online]. Available: https://mashable.com/article/robots-wildlife-photography. [Accessed: 28-Nov-2021].

[4] “Microchip.lk – one stop for all your electronics…,” *Microchip.lk*. [Online]. Available: https://microchip.lk/. [Accessed: 28-Nov-2021].

[5] “TRONIC.LK,” *Lankatronics.com*. [Online]. Available: https://lankatronics.com/. [Accessed: 28-Nov-2021].

## APPENDIX A

## Cost Estimation and Expenditure so far

| Component | Unit Price | Quantity | Price (LKR) |
|---|---|---|---|
| 68mm RC Car Tire Wheel | Rs. 300.00 | 4 | Rs. 1,200.00 |
| Liquid Crystal 16x2 Display Module | Rs. 300.00 | 1 | Rs. 300.00 |
| 2 Pin Switch | Rs. 15.00 | 2 | Rs. 30.00 |
| Thumb Joystick Module | Rs. 150.00 | 2 | Rs. 300.00 |
| RF 433MHz Transmitter/Receiver | Rs. 200.00 | 1 | Rs. 200.00 |
| GY-271 Electronic Triple Axis Compass Module | Rs. 700.00 | 1 | Rs. 700.00 |
| Membrane Keypad - 16 Key | Rs. 180.00 | 1 | Rs. 180.00 |
| HC-SR04 Ultrasonic Sensor Module | Rs. 200.00 | 6 | Rs. 1,200.00 |
| NEO-6M GPS Module | Rs. 1,850.00 | 1 | Rs. 1,850.00 |
| L298N DC Motor Driver Module | Rs. 350.00 | 2 | Rs. 700.00 |
| Servo Motor | Rs. 750.00 | 1 | Rs. 750.00 |
| Atmega32 Microcontroller | Rs. 550.00 | 2 | Rs. 1,100.00 |
| 11.1V 2200mAh 3S 25C Li-Po Battery | Rs. 2,800.00 | 1 | Rs. 2,800.00 |
| 12V 2000mAh Li-Po Battery | Rs. 1,750.00 | 1 | Rs. 1,750.00 |
| DC Gear Motor 12v 180 RPM | Rs. 950.00 | 4 | Rs. 3,800.00 |
| | | Total | Rs. 16,860.00 |

[4],[5]

# APPENDIX B

## Individual Contribution to the Project

### P.A.U.D. Herath - 204074M

- Studied about the GPS module.
- Studied about the way how the GPS signals generate and the way of GPS working.
- Learnt about the theory regarding the USART communication.
- Understood the theory behind the multiplexer.

### GPS module (NEO-6M)

The GPS module has a GPS antenna, battery, an EEPROM and a position fix LED indicator. The pins in the NEO-6M are Vcc, Tx, Rx and GND. Understood that the D0 and D1 pins must be used to take the serial data using the GPS module.

Specifications are,

- Operating Voltage: 2.7V - 3.6V
- Serial Baud Rate: 4800-230400(default 9600)
- Operating Current: 45mA

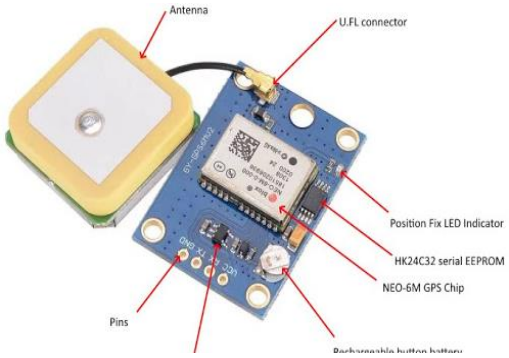



*Figure 21 GPS Module NEO-6M*

### Multiplexer

When taking the inputs related to the GPS module and the receiver, I have to use the same pins (D0, D1). Therefore, I used a multiplexer to carry this out.

### USART communication

To take the serial inputs from the GPS module, I used the USART method.

```
void UART_init(long USART_BAUDRATE);
unsigned char UART_RxChar();
void UART_TxChar(char ch);
void UART_SendString(char *str);

int main()
{
    char c;
    UART_init(9600);  //setting the baud rate to 9600Mbs

    UART_SendString("\n\t Echo Test ");
    while(1)
    {
        c = UART_RxChar();
        UART_TxChar(c);
    }
}

//Initialize the USART
void UART_init(long USART_BAUDRATE)
{
    UCSRB |= (1 << RXEN) | (1 << TXEN);  //Turn on transmission and reception
    UCSRC |= (1 << URSEL) | (1 << UCSZ0) | (1 << UCSZ1); // Use 8-bit character sizes
    UBRRL = BAUD_PRESCALE;      //Load lower 8-bits of the baud rate value
    UBRRH = (BAUD_PRESCALE >> 8);   // Load upper 8-bits
}
```

*Figure 22 USART sample code*

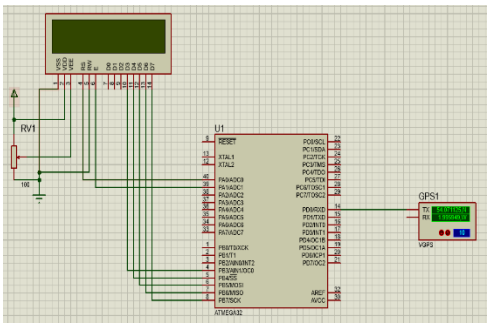



*Figure 23 GPS Module Proteus Schematic view*

**P.H.P. Jayathilaka 204087F**

- Studied about Ultrasonic sensors
- Studied about Display, Keypad, ADC, PWM
- Started studying about USART
- Started to study interfacing components with the ATmega32 microcontroller and how to connect components to the microcontroller.

*Figure 24 Ultrasonic sensor Physical view*


**Ultrasonic sensor (HC-SR04)**

I am using 6 ultrasonic sensors to avoid obstacles. There are 4 pins in the ultrasonic sensor. (VCC, TrigPin, EchoPin, GND)

Specifications of Ultrasonic sensor:

- Working Power: DC 5V; 15mA
- Working Frequency: 40Hz
- Accurate Range: 2cm – 40cm
- Measuring Angle: 15 degrees
- Trigger Input Signal: 10µS TTL pulse
- Echo Output Signal Input TTL lever signal and the range in proportion

```
motor.c  main.c  defines.c  ultrasonic.c  display
ultrazonic_distance   float ultrazonic_distance(Pin trigPi
 * Created: 11/1/2021 10:49:53 PM
 *  Author: Hansa Jayathilaka
 */

#include "../defines.h"


float ultrazonic_distance(Pin trigPin, Pin echoPin) {
    pin_mode(trigPin, OUTPUT);
    pin_mode(echoPin, INPUT_PULLUP);

    TIMSK = (1 << TOIE2); // Set timer2 to
    digital_write(trigPin, HIGH);
    _delay_us(10);
    digital_write(trigPin, LOW);
    unsigned long curr_time = time_us();
```

*Figure 25 Ultrasonic sensor Code*

**Gear Motors**

Due to the lack of 8 PWM pins for four motors, I had to couple motors as left and right. Then needed PWM pins reduced to 4. furthermore, I used only 1 PWM pin and a digital pin per side. It is configured by code.

Specifications of Gear motors:

- Working voltage: 12v DC
- Speed: 180RPM
- High torque motors
- Use L298N H-bridge motor drive

*Figure 27 High torque gear motor Physical view*

```
motor.c  main.c  defines.c  ultrasonic.c  display.c  ADC.c
motor.c   C:\Users\Hansa Jayathilaka\OneDrive - University of Mo
    if (reverse) {
        digital_write(DIRB, HIGH);
        PWM_write(PWM0B, 0xFF - speed);
    }
    else { // forward
        digital_write(DIRB, LOW);
        PWM_write(PWM0B, speed);
    }
}

void setM1Speed(int speed) {
    unsigned char reverse = 0;

    if (speed < 0) {
        speed = -speed; // make speed a positive quantity
        reverse = 1;    // preserve the direction
    }

    if (speed > 0xFF)
    speed = 0xFF;

    if (reverse) {
        digital_write(DIRA, HIGH);
        PWM_write(PWM0A, 0xFF - speed);
    }
    else { // forward
        digital_write(DIRA, LOW);
        PWM_write(PWM0A, speed);
    }
}

void drive(int m1Speed, int m2Speed) {
    setM1Speed(m1Speed);
    setM2Speed(m2Speed);
}
```

*Figure 26 Motor Controlling code*

**D.M.B.M. Dissanayake - 204047J**

- Studied about Joystick and RF Transmitter
- Studied about ADC
- Started to study interfacing components with the ATmega32 microcontroller and how to connect components to the microcontroller.

In our project, I used a joystick to get inputs related to the angle of servo motors. The microcontroller of the remote gets analog inputs from this joystick. We need knowledge about ADC on AVR ATmega32 to process that analog signal. To get this analog input, I had to use A0 pin in the ATmega32 microcontroller. Furthermore, I used a 433 MHz RF transmitter to transmit the above-processed data from the microcontroller.

**Thumb Joystick**

There are 5 pins in the joystick module.

VCC, GND, VRx, VRy, SW

Specifications:

- Operating Voltage: 5V
- Internal potentiometer value: 10kΩ
- Operating Temperature: 0 to 70℃

*Figure 28 Thumb Joystick Physical view*

There is no joystick module in proteus. Therefore, I used a variable resistor to connect with the microcontroller.

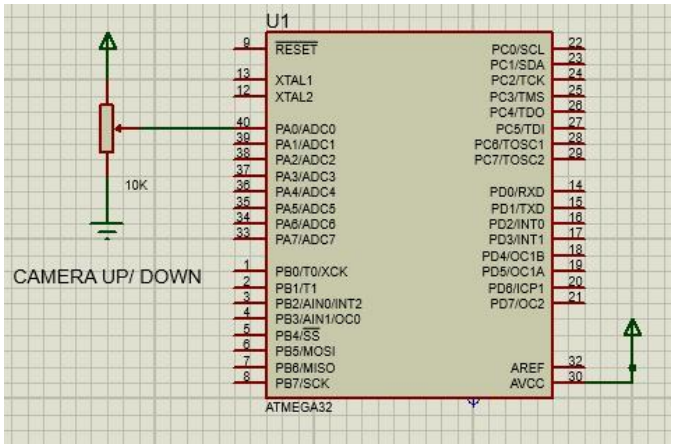



*Figure 30 Use variable resister for Joystick*

```
 * Joystick.c
 *
 * Created: 11/30/2021 11:37:35 AM
 * Author : ASUS
 */
#define CPU_F 8000000UL
#include <avr/io.h>
#include <stdio.h>

void ADC_init()
{
        DDRA = 0x00;  //make A pins as input
        ADCSRA = 0b10000111; //Enable ADC and fr/128
        ADMUX = 0b00100000; // choose Vref as AVCC; ADC channel 0

}

int ADC_Read(char channel)
{
    ADMUX = ADMUX | (channel & 0x0f); // Make input channel to read

    ADCSRA = ADCSRA | (1<<ADSC); //start conversion
```

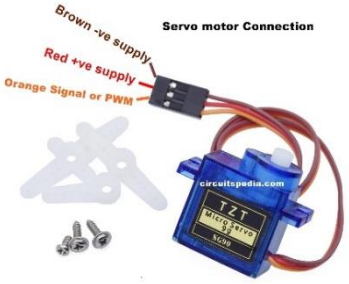
*Figure 29 ADC configuration and read*

**S.P.S.N. Pathirana 204150T**

- Studied about the Servo motors
- Studied about the Siren
- Started to learn about how they are working
- Started to learn about how they are interfacing with Atmega32 microcontroller

In this project, Servo motor is used to change the angle of the camera, according to the signal of joystick.

*Figure 31 Servo motor physical view*




**Servo Motor**

- Speed: 0.1 s
- Torque: 2.5 kg/cm
- Weight: 14.7 g
- Voltage: 4.8v - 6v

```
#define F_CPU 8000000UL // 8 MHz clock speed
#include <avr/io.h>
#include <util/delay.h>

int main(void)
{
    DDRC = 0x01; //Makes RC0 output pin
    PORTC = 0x00;

    while (1)
    {

        //Rotate Motor to 0 degree
        PORTC = 0x01;
        _delay_us(1000);
        PORTC = 0x00;

        _delay_ms(2000);

        //Rotate Motor to 90 degree
        PORTC = 0x01;
        _delay_us(1500);
        PORTC = 0x00;

        _delay_ms(2000);
```

*Figure 32 Servo motor sample code*

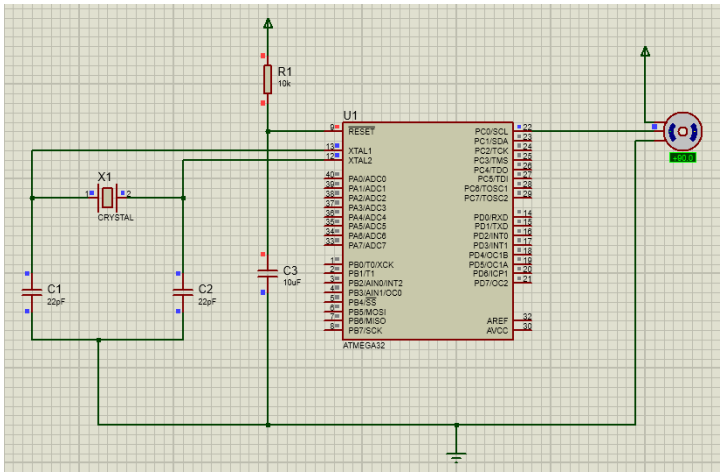



*Figure 33 Servo motor Proteus schematic view*

**Siren**

The siren is used for protecting the camera stand from the wild animals. When they come to attack it. The siren sounds according to the signal given by the siren button in the remote controller.

*Figure 35 Siren Physical view*

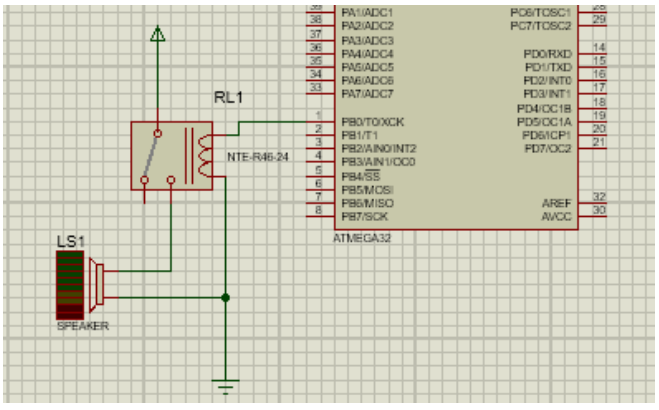



*Figure 34 Siren Proteus schematic view*

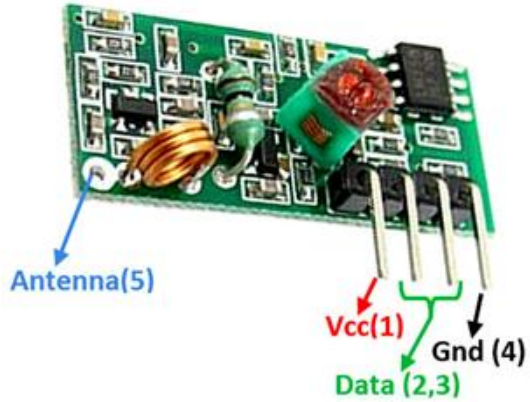
**A.M.D.B. Rathnayaka 204179N**

- Studied about the joystick
- Studied about the RF receiver
- Started to study about ADC
- Started to learn about how the components are working and interfacing with ATmega32 microcontroller

In our project, I have to give inputs to the ATmega32 microcontroller of the remote from joystick. Moreover, it gives inputs to the microcontroller of the host from the RF receiver.

**433 MHz Receiver**

*Figure 36 RF Receiver Physical view*




Receiver uses the USART communication to communicate with the Atmega32 microcontroller of the host.

Specifications:

- Frequency: 433 MHz
- Operating voltage: 5V
- Supply current: 3.5 mA

**Thumb Joystick**

There are 5 pins in the joystick module. And the pins in this module are VCC, GND, VRx, VRy, SW.

Specifications:

- Operating Voltage: 5V
- Internal potentiometer value: 10kΩ
- Operating Temperature: 0 to 70℃

```
 *
 * Parameter
 *   - None
 * Return
 *   - None
 */
void ADC_int(void) {
    ADCSRA = (1 << ADEN); // Enable pin; Enable ADC conversion
    ADCSRA |= (1 << ADPS2) | (1 << ADPS1) | (1 << ADPS0); // Set prescaler select bit; Division Factor 64

    ADMUX = (0 << REFS1) | (1 << REFS0); // Reference voltage selection; Select AVCC pin
    ADMUX = (0 << ADLAR); // ADC Left Adjust Result; Set to Right-Justified
}
```

*Figure 37 ADC initializing code*