FINAL REPORT

A

Project Report

ON

DISASTER ASSISTANCE ROBOT

Submitted By

Navkiran kamboj (730143)

Amanpreet Kaur (727884)

Gurjot Singh (721405)

Gurkirat Singh (730241)

Under the Esteemed Guidance of

Prof. Takis Zourntos

DEPARTMENT OF EMBEDDED SYSTEMS ENGINEERING DESIGN

LAMBTON COLLEGE IN TORONTO

1. ABSTRACT :-

Our proposed framework is an observing robot which can be of extraordinary guide to the rescuers who work in a disaster management. This robot is constrained by beagalboneblack and can be headed to areas where human invention is practically impossible or there exists risk to human life. This element can be accomplished utilizing OpenCV library to record the live data. In other words, from the live video we acquire from robot the rescuers can inspect a risky region from a faraway spot. What's more, the robot catches pictures from the area with the help of camera and detects any casualties around there and sends the picture and Location facilitates back to the client by email notice with the assistance of GPS module . The significant advantage of the robot is that it reduces the quantity of rescuers required in the accident region and it tends to be utilized for harm investigation.

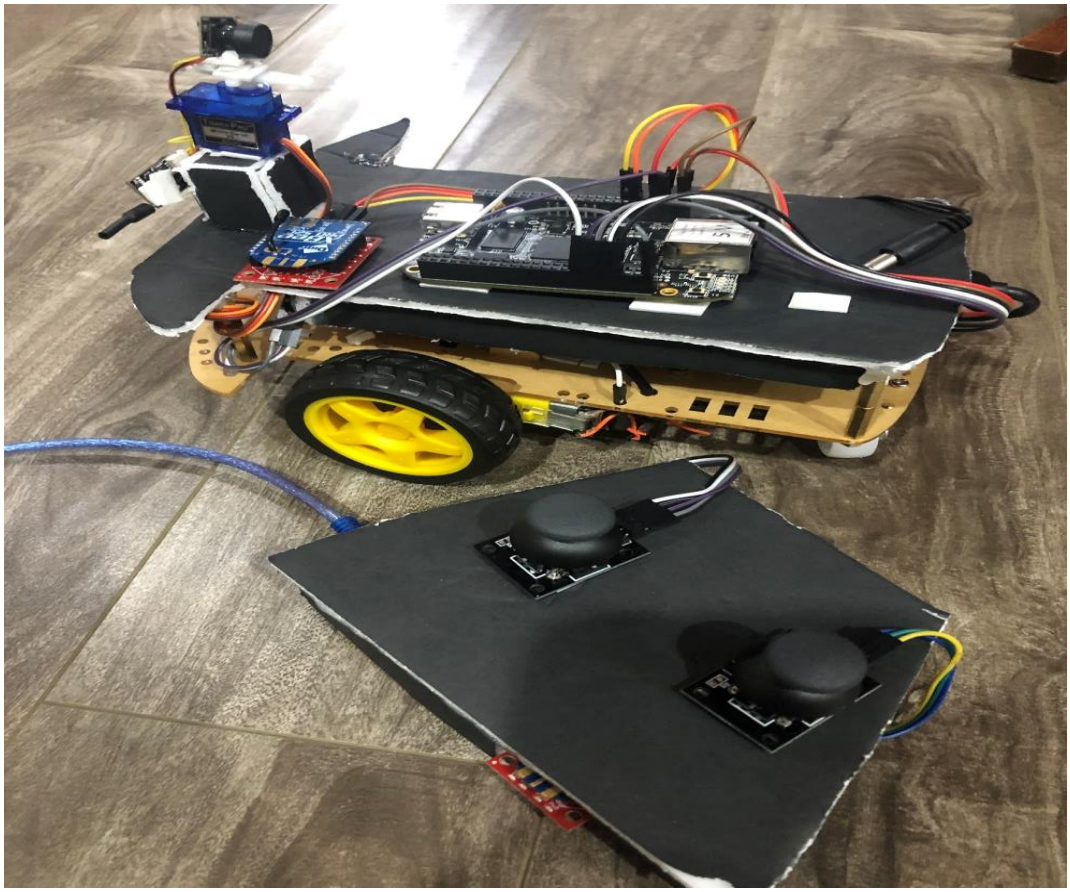
2. Introduction:-

2.1. Who are using this :-

There are a few common and man-made catamities happening all around the globe nowadays. The pollution and an Earth-wide temperature boost cause tremendous climate changes which in turn trigger natural events removing human lives . Also, world is experiencing a lot from terroriest attacks, serious accidents in atomic force stations and in transportation part. So as to handle these issues every nation has there on disaster supervisory group and equipment's. This procedure is said to be very extremely complex and non-direct undertakings that includes various procedure of dynamic coordination and participation between different on-screen characters and establishments to achieve approaches, techniques and capacities too manufacture limits during all times of calamities the executives cycle to restrict the impacts of dangers, save lives, improve employments and secure beneficial resources and system. Prepared disaster management experts can utilize this robot from a distant area and monitor the zone under risk. This robot is extremely helpful in cases

of toxic gas out breaks and earth quakes since it is risky for the rescuers to look at the affected spots without anyone else. They can utilize this help robot to screen the zone and send help whenever require.

Hardware:



2.2.How it works:-

The significant objective is to equip the robot with a fpv camera with transmitter, xbee module IS2, beaglebone black,servo motor and motor driver to make it move at various spot with the help of DC motors while it is controlled from a faraway spot with the help of remote. The camera catches live occasions as the robot goes through the disaster place and this feed can be analyzed by experts from far off spots. On the off chance that the rescuers see any human or creature nearness through the camera quickly, they can send help to the area where the robot is available.

Though, Pocket beagle bone which gets information from fpv Camera through xbee, and move it with servo motor which rotate the camera at every 180 degree. Along these lines, when fpv camera recognizes any articles, it sends x, y axis, and height, and width of item to beagle bone. In this way, by figuring the distance, and angle of object pocket beagle bone sends that information to camera.

2.3 Description of our problems:-

Initially,beagalbone was not updated . we updated it with 9.5 version

We were trying to write our code manually for servo motor because there was no c library for beagalbone to generate pwm (pulse width modulation) own.

Actually, our code needs libiobb.a for compilation.Generally, we can write liobb in last of compilation command and it did work.Unfortunately, eclipse was not supporting. So, we download new library libobb.

At the end, Iobb.h library was working on GPin,out. UART PWM not working then bash code wrote for unable uart4 by using file accessing directly .

3 Hardware:-

- Black Beagle bone

- FPV camera with camera
- Motor Driver
- Joystick
- Battery charger
- Servo Motors
- DC Motors
- Batteries
- Jumper Wires
- Wireless Xbee Communications

3.1. Beagle bone Black: Beagle bone is an open source, and ultra-tiny. It has powerful 1GHz AM3358 powered Linux single board. Beagle bone is small in size, and has every feature. In our project, FPV Cam sends data to beagle bone through SPI serial communication using SPI0 port, and then beagle bone transfers that data to Motor driver through UART communication.

3.2. FPV Camera: Wireless FPV Camera is a good device to give vision to a robot. It is smaller, faster, and very capable camera. FPV camera detect different objects, and tell your robot what to do, for example in which direction robot should move. In our project, FPV camera search for different objects, and when it detects any objects it sends data to black beagle bone by using SPI interfacing.

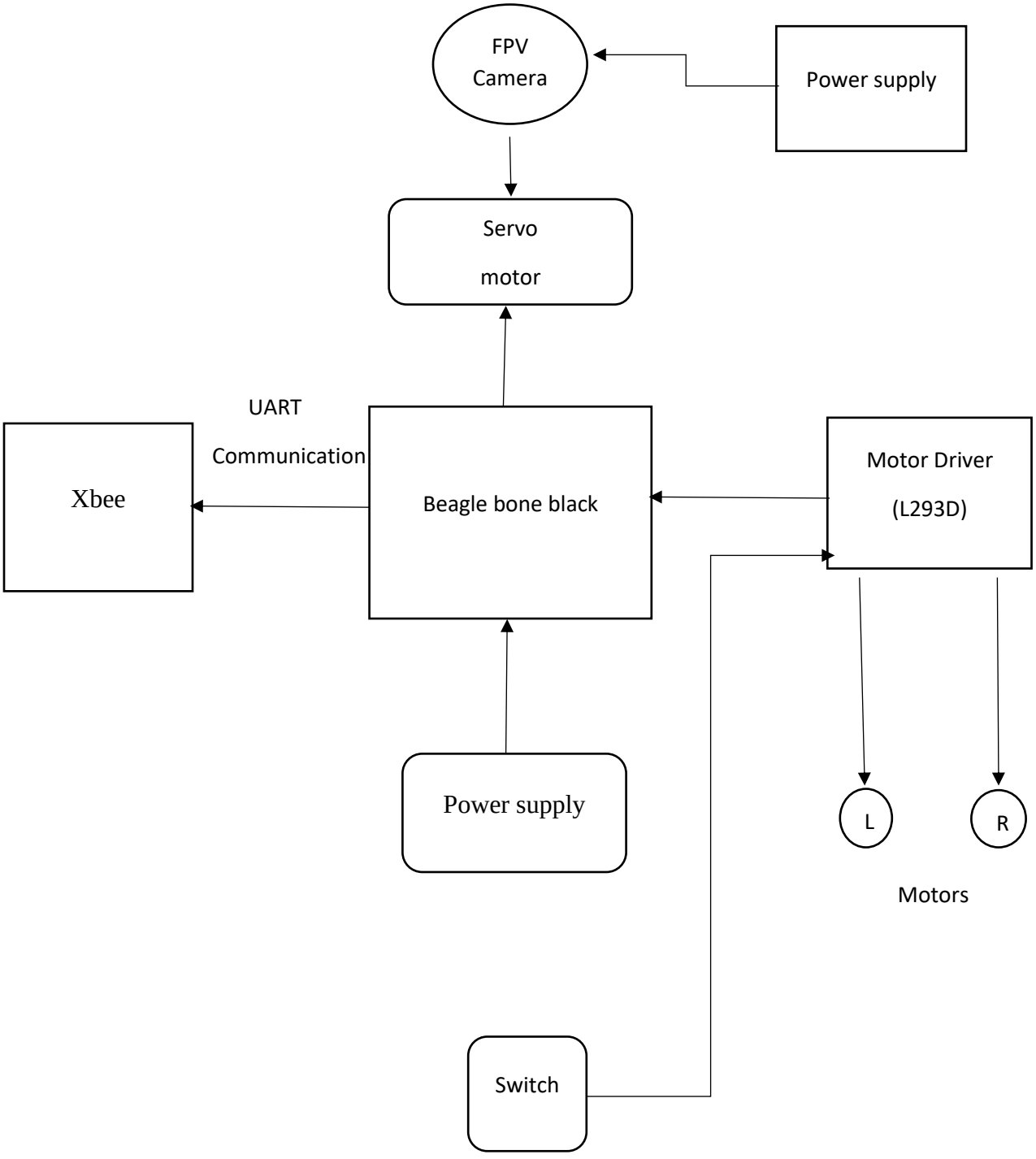
3.3. Motor Driver: We used motor driver to control our robot motors. Its four inputs are connected to Beagle bone and four outputs are connected to left motor, and right motor. Motors Driver works at 6V which is provided by voltage regulator.

3.4. Joysticks: This is able to send data to beagle bone via Xbee communication in X and Y data separately.

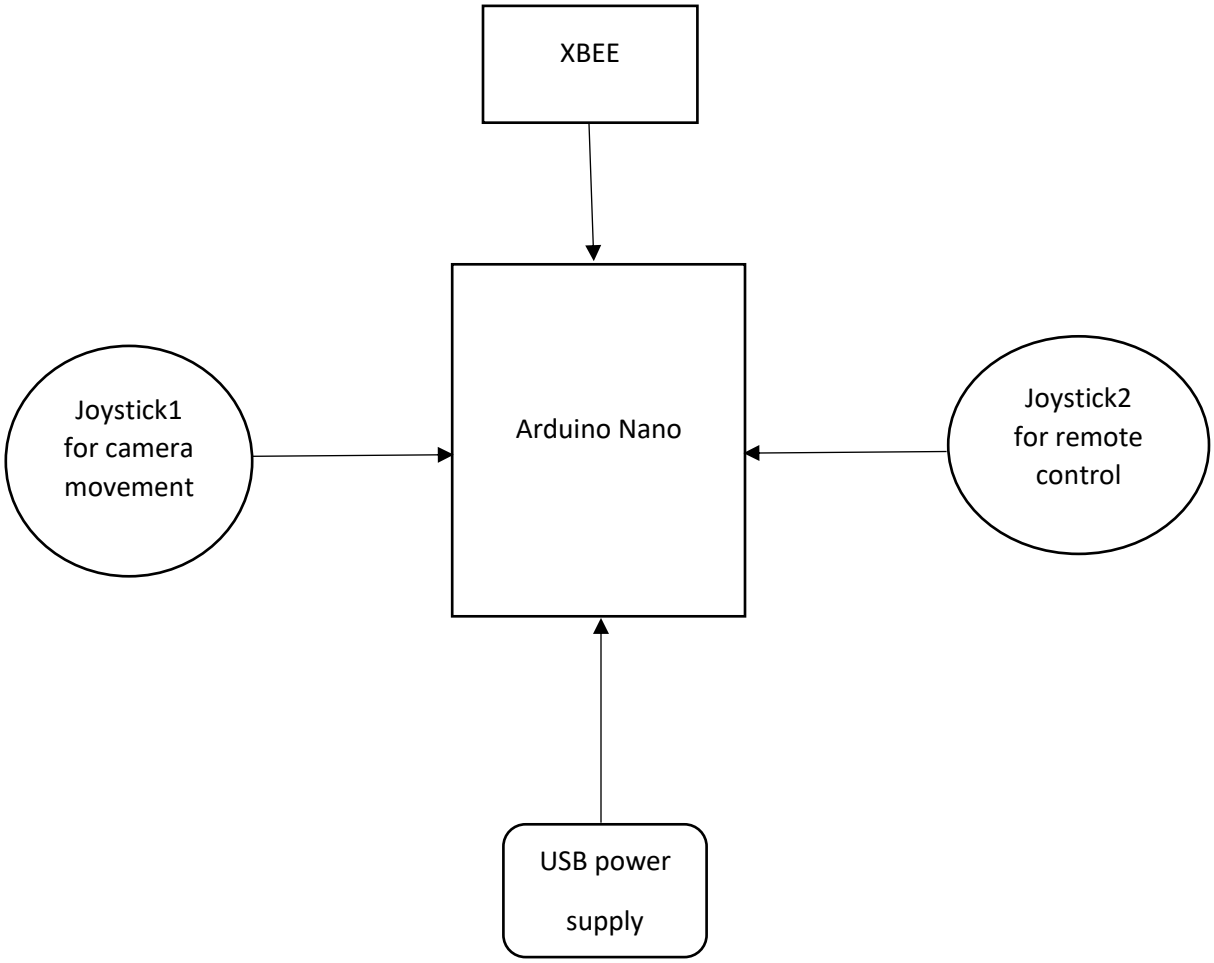
3.5 Servo motors: This is using for rotating the camera at different angle each is 90 degree. It helps to rotate the camera left, right, forward and downward.

3.6 Xbee Communication module: XBee is a RF (radio frequency) module which is used as wireless communication .It helps to transfer the data from remote to beagle bone.

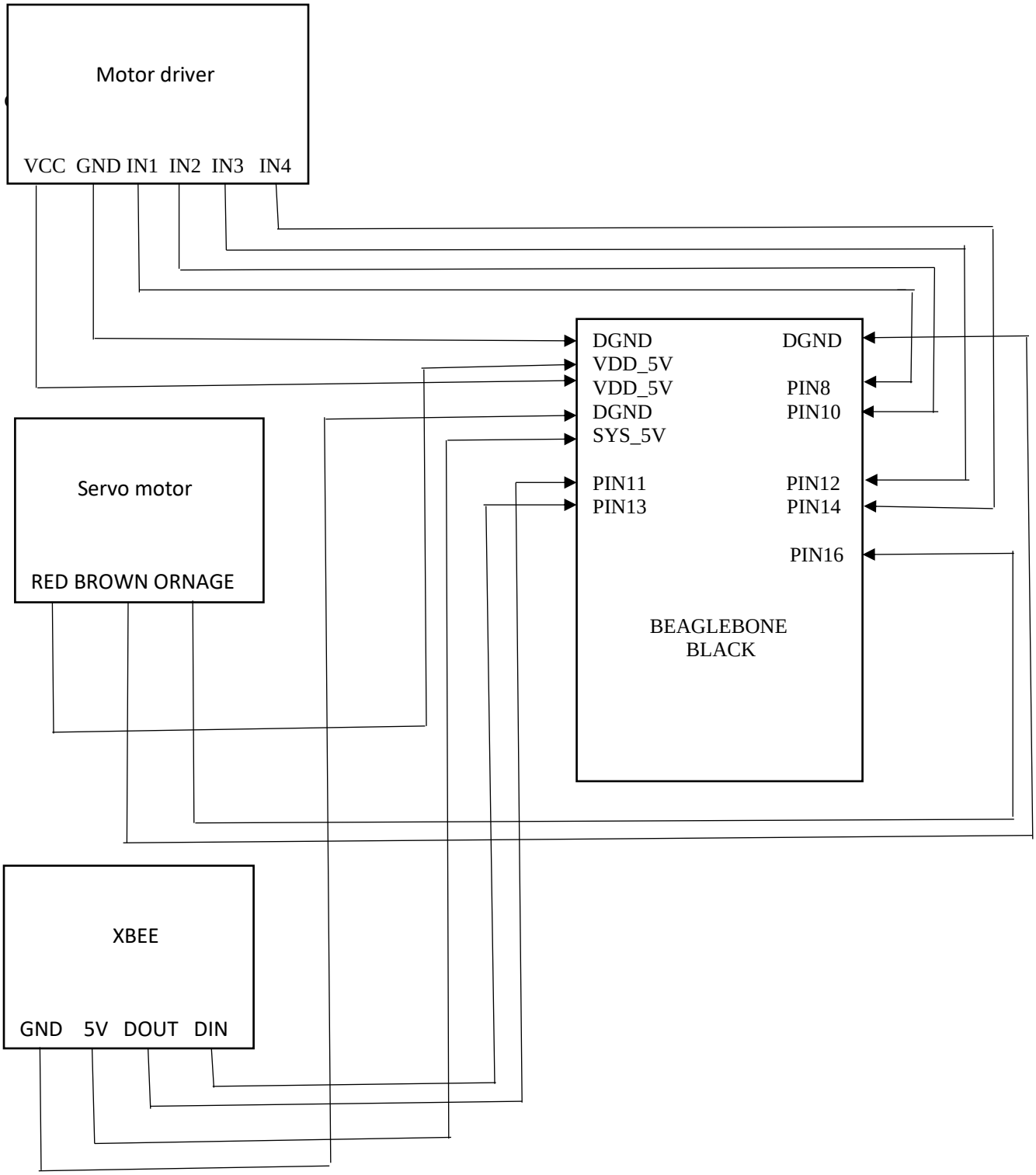
4. Block Diagram:



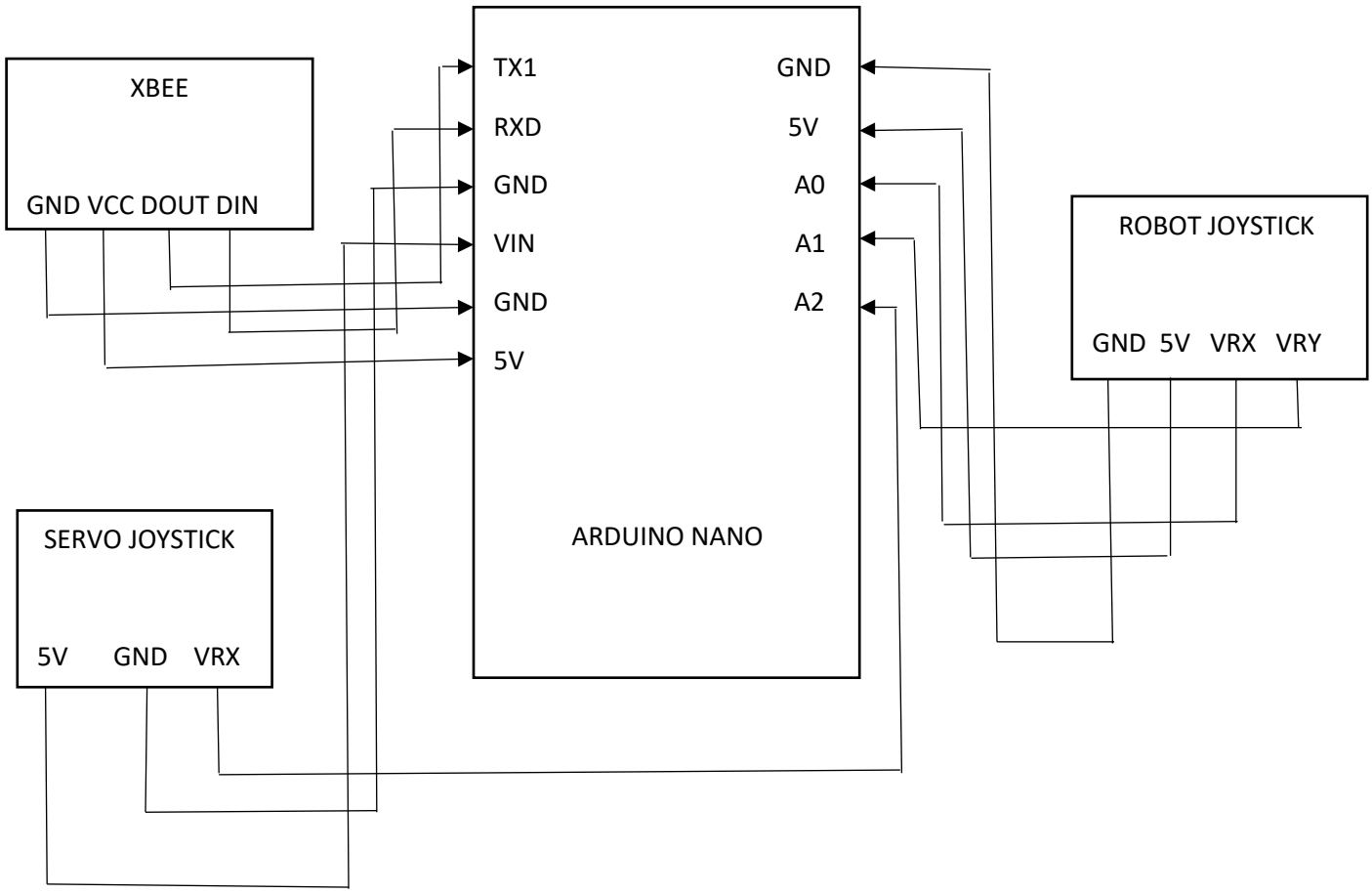
4.1. Remote block diagram



4.2. Servo motor, Motor driver and Xbee with Beaglebone Black Communication Diagram:


Motor driver
VCC GND IN1 IN2 IN3 IN4
Servo motor
RED BROWN ORNAGE
XBEE
GND 5V DOUT DIN
DGND
VDD_5V
VDD_5V
DGND
SYS_5V
PIN11
PIN13
DGND
PIN8
PIN10
PIN12
PIN14
PIN16
BEAGLEBONE
BLACK

4.3 ARDUINO NANO, XBEE, AND JOYSTICKS COMMUNICATION DIAGRAM:



5. Software Description:-

5.1.Main code:

```
#include <iostream>    // iostream is the header file which contains all   the functions of program inputs/outputs
#include <stdio.h>    // Standard C input Output Library
#include <unistd.h>    // defines miscellaneous symbolic constants and types, and declares miscellaneous functions
#include <string.h>      // C Library for various String Operations
#include <termios.h>   // Contains the definitions used by the terminal I/O interfaces
#include <stdlib.h>     //  General standard Library for various Functions
#include <sys/types.h>   //  definitions for types like size_t , ssize_t
#include <sys/stat.h>   //  header defines the structure of the data returned by the functions fstat(), lstat(), and stat(), give file size
#include <fcntl.h>   // FIle control, Open, close
#include "iobb.h"    // A header library to control GPIOs of Beaglebone
#include <stdio.h>     // Standard C input Output Library
#include <time.h>     //  Time Library

int servo_angle(int angle, int Servopin);  //Defining a function

int thres[] = {400,600,400,600,250,450};
// x_low, x_high, y_low, x_high, c_low, c_high

const int IN1 = 8;   // Pins where Moto driver inputs are connected, Use for controlling motors
const int IN2 = 10;
const int IN3 = 12;
const int IN4 = 14;

const int ServoPin = 16; // Pin where servo is connected

int serAngle =90; // Initial Servo Angle

int sigCount = 0;

 int main (void) // Main Function
{

   iolib_init();  //Initializing the iobb library

   iolib_setdir(8, IN1, DigitalOut); //Setting Pin direction of a specific pin of a specific port
   iolib_setdir(8, IN2, DigitalOut);
   iolib_setdir(8, IN3,  DigitalOut);
   iolib_setdir(8, IN4,  DigitalOut);
   iolib_setdir(8, ServoPin, DigitalOut);
```

```
printf("Program Started\n"); // A simple print

servo_angle(serAngle, ServoPin); // Set the servo initially at 90 degrees

        int file, i; // Variable integers
        char receive[100]; // declare a char array for receiving data
        char buf[20]; // A buffer char array to store temporary data
        size_t nbytes; //size_t is an unsigned integer data type used for storing size
        size_t bytes_written; // //size_t is an signed integer data type used for storing size


        if ((file = open("/dev/ttyO4", O_RDWR))<0) // Try opening file in Read Write mode
        {
        printf("UART: Failed to open the file.\n"); //A message Print
        return 0;
        }

        //
        struct termios options; // the termios structure is vital
        tcgetattr(file, &options); // sets the parameters associated with file

        // Set up the communications options:
        // 9600 baud, 8-bit, enable receiver, no modem control lines
        options.c_cflag = B9600 | CS8 | CREAD | CLOCAL;
        options.c_iflag = IGNPAR | ICRNL; // ignore parity errors, CR -> newline
        tcflush(file, TCIFLUSH); // discard file information not transmitted
        tcsetattr(file, TCSANOW, &options); // changes occur immediately

        strcpy(buf, "This is a UART test\n"); // Copy a string in buf char array
        nbytes = strlen(buf); // Store size of buf array in nbytes


        while (1) // Infinite loop
        {
        //bytes_written = write(file, buf, nbytes); // Writing data, we don't need this
int i = 0;
int bytes_read = 0;

bytes_read = read(file,receive,100); // Read the file and store the data in receive , read 100 bytes max
printf("\n\nBytes Received - %d",bytes_read); // Print how many bytes was received
printf("\n");
if(bytes_read > 10)  //If no. of bytes are read is more than 10
{
for(i=0;i<bytes_read;i++)
{
printf("%c",receive[i]);
```

```
    }
    printf("\n----------------------------------------\n\n");


        sleep(1);


char a[25]; //A Char array for holding temporary data
char data[4]; //A char array to hold data

memcpy(a,receive,bytes_read);// Copying data from rec to a
a[bytes_read+1]='\0'; //Adding a Null character at the end of array

printf("Received Data is %s ",a); // Print Statement to print received data
printf("\n");

char *xxpos; //A char array
xxpos = strchr(a,'X'); //To get the first occurance of 'X' in 'a'
int xpos = int(xxpos-a); // To get absolute index of 'X' in 'a'
printf("Position of X : %i\n",xpos); //Print position of X

char *yypos; //A char array
yypos = strchr(a,'Y'); //To get the first occurance of 'Y' in 'a'
int ypos = int(yypos-a); // To get absolute index of 'Y' in 'a'
printf("Position of Y: %i\n",ypos); //Print position of Y


char *ccpos; //A char array
ccpos = strchr(a,'C'); //To get the first occurance of 'C' in a
int cpos = int(ccpos-a); // To get absolute index of 'C' in a
printf("Position of C: %i\n",cpos); //Print position of C



char *hypppos; //A char array
hypppos = strchr(a,'H'); //To get the first occurance of 'H' in a
int hyppos = int(hypppos-a); // To get absolute index of 'H' in a
printf("Position of H: %i\n",hypppos); //Print position of H



char *unddpos; //A char array
unddpos = strchr(a,'U'); //To get the first occurance of 'U' in a
int undpos = int(unddpos-a); // To get absolute index of 'U' in a
printf("Position of U: %i\n",undpos); //Print position of U
```

```
memset(data,'\0',sizeof(data)); //Make all the characters stored in data as null
for(int i=xxpos+2;i<hypppos;i++)  // A for loop to copy X data
{
printf("%c",a[i]); //Print the character
data[i-(xxpos+2)] = a[i]; //Store the data in data variable
}
printf("- X Value \n"); //  Print
int x_data = atoi(data);//Convert Character array data to integer
if(x_data>1024) x_data = x_data/10;
printf("Integer value of X %d\n",x_data); //Print integer converted data


memset(data,'\0',sizeof(data)); //Make all the characters stored in data as null
for(int i=yypos+2;i<unddpos;i++) // A for loop to copy Y data
{
printf("%c",a[i]); //Print the character
data[i-(yypos+2)] = a[i];  //Store the data in data variable
}
printf("- Y Value \n"); //  Print

int y_data = atoi(data); //Convert Character array data to integer
if(y_data>1024) y_data = y_data/10;
printf("Integer value of Y %d\n",y_data); //Print integer converted data


memset(data,'\0',sizeof(data)); //Make all the characters stored in data as null
for(int i=ccpos+2;i<strlen(a)-1;i++) // A for loop to copy Y data
{
printf("%c",a[i]); //Print the character
data[i-(ccpos+2)] = a[i];  //Store the data in data variable
}
printf("- C Value \n"); // General Print

int c_data = atoi(data); //Convert Character array data to integer
if(c_data>1024) c_data = c_data/10;
printf("Integer VAlue of C %d\n",c_data);  //Print integer converted data


if (x_data > thres[0] and x_data < thres[1] and y_data > thres[2] and y_data < thres[3]) // If any value is
beyond threshold
  {
        printf("STOP\n"); // Print the Current State of Robot
        pin_low(8,IN1); // Make all pins low, stops both motor
        pin_low(8,IN2);
        pin_low(8,IN3);
```

```
        pin_low(8,IN4);
}

else
{
        if(x_data >= thres[1])
        {
        printf("FORWARD\n"); // Print the Current State of Robot
        pin_high(8,IN1); // Make both motors run in forward direction
        pin_low(8,IN2);
        pin_high(8,IN3);
        pin_low(8,IN4);
        }
        else if(x_data <= thres[0])
        {
          printf("BACKWARD\n"); // Print the Current State of Robot
          pin_low(8,IN1); // Run both motors run in opposite direction to move motor backward
          pin_high(8,IN2);
          pin_low(8,IN3);
          pin_high(8,IN4);
        }
        else if(y_data >= thres[3])
        {
          printf("RIGHT\n"); // Print the Current State of Robot
          pin_low(8,IN1); // Make 1 motor off and 1 on to move robot in right direction
        pin_low(8,IN2);
          pin_high(8,IN3);
          pin_low(8,IN4);
        }
        else if(y_data <= thres[2])
        {
        printf("LEFT\n"); // Print the Current State of Robot
        pin_high(8,IN1);  // Make 1 motor on and 1 off to move robot in left direction
        pin_low(8,IN2);
        pin_low(8,IN3);
        pin_low(8,IN4);
        }
}


if(c_data > thres[4] and c_data < thres[5])
{
        pin_low(8,ServoPin); // Make the servo pin low
}

else
{
```

```
            if(c_data <= 250)
            {
              printf("Camera Moving Left\n"); // Print the Current State of Servo

              serAngle -= 10;
              serAngle = servo_angle(serAngle,ServoPin);
              tcflush(file, TCIFLUSH); // discard file information not transmitted


            }
            if(c_data >= 450)
            {
              printf("Camera Moving Right\n");  // Print the Current State of Servo

              serAngle += 10;
              serAngle = servo_angle(serAngle,ServoPin);
              tcflush(file, TCIFLUSH); // discard file information not transmitted

            }
    }


    }
    else
    {
    sigCount++; // A integer to track for how many loops we didn't receive signal from Remote

    if(sigCount > 20) // If no signal received for 20 loops
    {
            printf("STOP\n");  // Print the Current State of Robot
            pin_low(8,IN1); //Making all pins low, stopping both motors
            pin_low(8,IN2);
            pin_low(8,IN3);
            pin_low(8,IN4);
            sigCount =0; //make the variable zero
    }
    iolib_delay_ms(100); //A delay of 100 ms
    }
            }
            iolib_free(); // Free the GPIOs
            close(file); // Close the file
}

 int servo_angle(int angle, int Servopin) // Function for moving servo to angle and the Pin servo
connected to
 {
```

```
 int delay = 0;  // An integer to store delays

 if(angle>=180) angle = 180; //If angle is greater than 180, limit it 180
 if(angle <= 0) angle = 0; //If angle is greater than 0, limit it 0

 delay =( (5.56 * angle)+1000); // A formula to calculate delay for Servo motor
// We got the delay and other details from Servo motor (SG90) datasheet

 printf("Delay %d \n",delay);  // Print the calculated delay

 for(int i=0;i<50;i++)  // A for loop running 50 times for setting angle of Servo
 {
 pin_high(8,Servopin); // Making the servo connected pin high
 usleep(delay); //Delay in microseconds which  calculated above
 pin_low(8,Servopin); // Making the servo connected pin low
 usleep(20000-delay); //Delay in microseconds which calculated above subtracted by 20000

 }

 iolib_delay_ms(10); // Small delay

 return(angle); //Return the current angle


 }
```

5.2. **Remote code:**

```
#include <SoftwareSerial.h> // Library for creating virtual UART port


int x = 0; // Variables to store values
int y = 0;
int x_cam = 0;


SoftwareSerial mySerial(10, 11); // Defining Virtual UART port at pin 10 and 11
// The above UART port will be used for communication with XBEE module

String data; // String Value for final output
```

```
void setup() {

  Serial.begin(9600);// Starting Serial UART communication
  mySerial.begin(9600);// Starting Serial UART communication with XBEE

  pinMode(A0,INPUT); // X
  pinMode(A1,INPUT); // Y


  pinMode(A2,INPUT); // X - 2nd Joystick

  Serial.println("Remote Joystick");  // A simple Print


}

void loop() {
  x = analogRead(A0); // Reading Values from both the Joysticks
  y = analogRead(A1);

  x_cam = analogRead(A2);

  data += "X:"; // Storing the values in a proper String Format
  data += (String)x;
  data += "-Y:";
  data += (String)y;
  data += "_C:";
  data += (String)x_cam;

  Serial.println(data);  // Printing the final String on Serial port

  Serial.println("Data Sent to Router");// SImple Print

  mySerial.println(data); // Send the data to XBEE


  delay(200); // Small delay to make sure data transmission is completed and XBEE is not overwhelmed
by incoming data

  data = ""; // Erase the string data


  /*
  if (mySerial.available()) {
          Serial.write(mySerial.read());
  }
```

```
 if (Serial.available()) {
        mySerial.write(Serial.read());
 }
*/


}
```

6. Demo-Day Experience:

The robot works properly on demo day. Although we faced some problems before that we were looking for 12-volt adapter but because of Covid we were not able to find anywhere so we made our own adapter for power supply. After When we connected the power to motors, it suddenly turned off when code loads into beagle bone. But we did resolve the issue, and motors starts working properly.

7. YouTube Link:

https://www.youtube.com/watch?v=yOpe1o5dYxk&feature=youtu.be

8. Github:

- For Robot code:

  https://github.com/Amanp19/final-project/tree/master/Amanpreet%20kaur/final_code

- For Remote code:
  https://github.com/Amanp19/final-project/tree/master/software/remote

9. References:

https://www.sciencebuddies.org/science-fair-projects/project- ideas/Robotics_p015/robotics/search-and-rescue-robot

https://beagleboard.org/Support/bone101

https://www.digi.com/xbee

https://en.wikipedia.org/wiki/XBee

https://en.wikipedia.org/wiki/First-person_view_(radio_control)