NATIONAL INSTITUTE OF TECHNOLOGY
TIRUCHIRAPPALLI

RMI
ROBOTICS AND MACHINE INTELLIGENCE

SUMMER PROJECT

Leader Follower Robot

By:
Gudapati Nitish
111117036
Mechanical Engineering

Acknowledgement

A project is a chance for us to learn, innovate and improve our scope of knowledge, so that we are better equipped to tackle real life problems. I wish to thank the RMI team for entrusting this project to me, hence encouraging me to explore beyond my limits.

The project was a complete learning process, and kept me continuously engaged throughout the summer break. It helped me get back in touch with Python, and learning new concepts.

I would like to specially thank my mentors Sri Krishna, Vidyaa and Dhanavel for their indescribable guidance throughout the whole project.

Contents

1 Project Introduction

1.1 Basic Task

This task involves replication of the path of the Leader robot by the follower robot.

1.2 Advanced Task

The advanced task is localisation using odometer readings derieved from the encoder in Leader Robot and the Follower robot has to move to that location.

2 Hardware Requirements

The hardware requirements are three Arduinos and three XBee modules (two for the bots and one for interfacing with the laptop), one IR Array, four DC Motors with each one fitted with wheel encoders (IR Tachometer), two dc motor drivers, two metal chassis

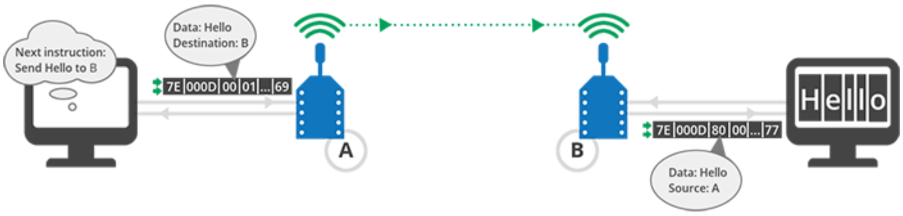
3 Software Requirements

The softwares required are Arduino IDE, XCTU, Python 2, PySerial and VPython Library.

4 Description

4.1 XBee

In-order to send the processed data between the robots and computer, Xbee devices were used. Xbees are the devices which are used for setting up wireless communication. These devices work on Zigbee wireless protocol. Zigbee was created and validated by an alliance of more than 300 leading semiconductor companies. It is based on IEEE 802.15.4 standard for wireless communication. This protocol is mainly developed for simple connectivity, lower battery consumption and provides minimum latency. For Xbees, the 2.4 Ghz band range is the most popular and widely used communication band. Another advantage of Zigbee protocol is that the Xbees can be designed to form a mesh network. Using mesh network, communication between Xbees which are located farther than their range can communicate easily. They can also be set up as point to multi-point network topology.

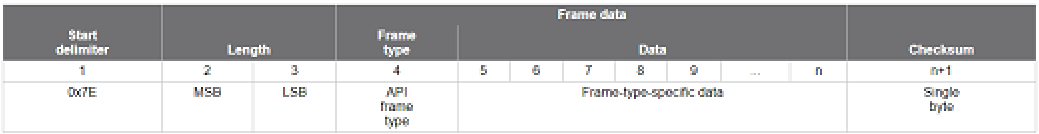


XBee works in two modes - transparent mode (AT) and application program interface mode (API). API mode provides a structured interface where data is communicated through the serial interface in organized packets and in a determined order. This enables you to establish complex communication between devices without having to define your own protocol. By default, XBee devices are configured to work in transparent mode: all data received through the serial input is queued up for radio transmission and data received wirelessly is sent to the serial output exactly as it is received, with no additional information. To read or write the configuration of an device in Transparent mode, you must first transition the device into Command mode.If a device needs to transmit messages to different devices, you must update its configuration to establish a new destination. The device must enter Command mode to set up the destination. A device operating in Transparent mode cannot identify the source of a wireless message it receives. If it needs to distinguish between data coming from different devices, the sending devices must include extra information known by all the devices so it can be extracted later. To do this, you must define a robust protocol that includes all the information you think you need in your transmissions.

To minimize the limitations of the transparent mode, devices provide an alternative mode called Application Programming Interface (API). API mode provides a structured interface where data is communicated through the serial interface in organized packets and in a determined order. This enables you to establish complex communication between modules without having to define your own protocol. API mode provides a much easier way to perform the actions listed above. Since the data destination is included as part of the API frame structure, you can use API mode to transmit messages to multiple devices. The API frame includes the source of the message so it is easy to identify where data is coming from.It can receive success/failure status of each transmitted packet and obtain the signal strength of any received packet. In this project, API mode is used along with ZB TH reg firmware.

4.1.1 API frame structure

The structured data packets in API mode are called frames. They are sent and received through the serial interface of the device and contain the wireless message itself as well as some extra information such as the destination/source of the data or the signal quality. When a device is in API mode, all data entering and leaving the module through the serial interface is contained in frames that define operations or events within the device. An API frame has the following structure:



			Frame data								
Start delimiter	Length		Frame type	Data							Checksum
1	2	3	4	5	6	7	8	9	...	n	n+1
0x7E	MSB	LSB	API frame type	Frame-type-specific data							Single byte

Any data received through the serial interface prior to the start delimiter is silently discarded by the XBee. If the frame is not received correctly, or if the checksum fails, the data is also discarded and the module indicates the nature of the failure by replying with another frame.

Start delimiter:

The start delimiter is the first byte of a frame consisting of a special sequence of bits that indicate the beginning of a data frame. Its value is always 0x7E. This allows for easy detection of a new incoming frame.

Length

The length field specifies the total number of bytes included in the frame data field. Its two-byte value excludes the start delimiter, the length, and the checksum.

Frame data

This field contains the information received or to be transmitted. Frame data is structured based on the purpose of the API frame.

Frame type is the API frame type identifier. It determines the type of API frame and indicates how the information is organized in the Data field.

Data contains the data itself. The information included here and its order depends on the type of frame defined in the Frame type field.

Checksum:

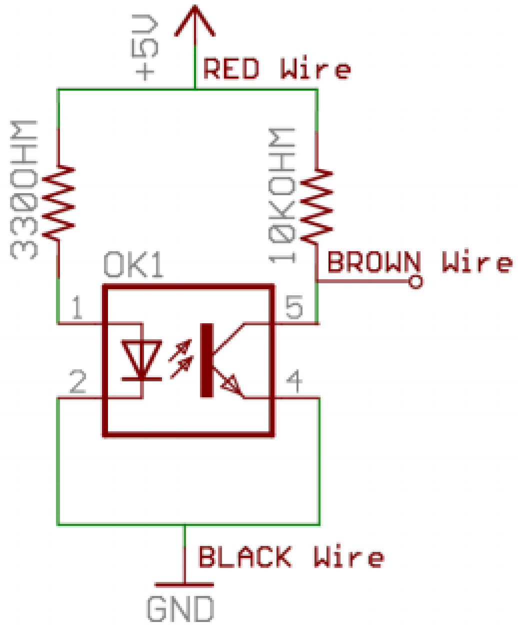
Checksum is the last byte of the frame and helps test data integrity. It is calculated by taking the hash sum of all the API frame bytes that came before it, excluding the first three bytes (start delimiter and length).

Note: Frames sent through the serial interface with incorrect checksums will never be processed by the module and the data will be ignored.

Calculate the checksum of an API frame

Add all bytes of the packet, excluding the start delimiter 0x7E and the length (the second and third bytes). From the result, keep only the lowest 8 bits. Subtract this quantity from 0xFF.

4.2 Wheel Encoder (IR Tachometer)



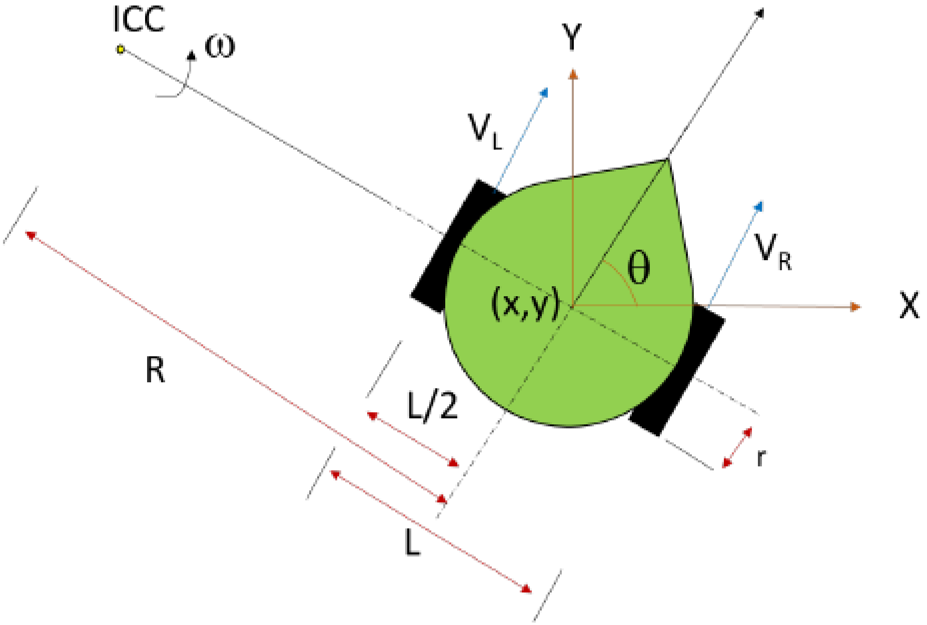
4.3 PID Controller

Proportional, Integral, Derivative PID controller is a feedback controller that helps to attain a set point irrespective of disturbances or any variation in characteristics of the plant of any form. It calculates its output based on the measured error and the three controller gains; proportional gain Kp, integral gain Ki, and derivative gain Kd. The proportional gain simply multiplies the error by a factor Kp. This reacts based on how big the error is. The integral term is a multiplication of the integral gain and the sum of the recent errors. The integral term helps in getting rid of the steady state error and causes the system to catch up with the desired set point. The derivative controller determines the reaction to the rate of which the error has been changing. In most of the systems, it is not necessary to use the derivative part of the PID, hence in this project, only PI controller has been designed and used. The final output of the controller (U) is calculated using the following equation:

$$U = Kp * e(t) + Ki \int e\, dt + Kd \frac{de}{dt}$$

Parameter Increase	**Rise time**	**Overshoot**	**Settling Time**	**Steady-state error**
Kp	↓	↑	Small Change	↓
Ki	↓	↑	↑	Great reduce
Kd	Small Change	↓	↓	Small Change

4.4 Odometry



c = (2*PI)/((Ticks per revolution * dt)/1000000.0)
wl = (leftTicks-leftTicksPrev)*c
wr = (rightTicks-rightTicksPrev)*c
Vl = wl*RADIUS
Vr = wr*RADIUS
V = (Vr + Vl)/2.0
w = (Vr - Vl)/LENGTH
thetaNext = theta + dt*w
xNext = xc + (dt*v*cos(thetaNext))/10.0
yNext = yc + (dt*v*sin(thetaNext))/10.0
distance = sqrt(xc*xc + yc*yc)

5 Basic Task

This task involves replication of the path of the Leader robot by the follower robot. There are two ways of solving the problem. Firstly, The leader robot tracks a line and and sends the rpm values of each of its wheel. The corresponding wheels of the follower are set to run at the same rpm as that of the leader using PID control algorithm. Thereby, the follower replicates the leader’s path. The second way is by sending the velocity of the bot and its orientation. In this project the first method is implemented. First, the Leader robot tracks a line using IR Array Sensor. Simultaneously, it calculates the wheel rpm and it’s x,y coordinates and orientation. All the above data along with the wheel direction is sent as a packet through XBee.

The follower uses proportional controller to set the rpm and direction of each of the wheels. It then sends the x,y,theta of both the leader and the follower bots to the Laptop Xbee. The paths are then plotted in Python Serial.

5.1 Source Code

5.1.1 Leader

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include <XBee.h>

XBee xbee = XBee();
ZBTxStatusResponse txStatus = ZBTxStatusResponse();

#define RADIUS 25.75
#define LENGTH 166
#define TICKSPERREV 30

#define UINT_MAX 4294967295

XBeeAddress64 addr64_Broadcast  = XBeeAddress64(0x00000000, 0x0000FFFF);
XBeeAddress64 addr64_Follower   = XBeeAddress64(0x0013A200, 0x41517A82);
XBeeAddress64 addr64_Laptop   = XBeeAddress64(0x0013A200, 0x415177B1);

int eleft,left,centre,right,eright;
int rm1=8, rm2=9, rme=10, lm1=12, lm2=13, lme=11;

int lsp=155,rsp=155;
long line=0,pre=0,start=0;
volatile int lpos=0,rpos=0;
float rrot=0,lrot=0;
volatile unsigned long lctr = 0, rctr = 0, lpulse = 0, rpulse = 0;
long leftTicks = 0, rightTicks = 0, leftTicksPrev = 0, rightTicksPrev = 0;
volatile unsigned long lrpm, rrpm, duration, prev = 0, cur = 0, t = 0;

unsigned long prevPositionComputeTime = 0, prevWheelComputeTime = 0, prevIntegrationTime = 0;
double prevX = 0, prevY = 0, xc = 0, yc = 0, theta = 0, wl = 0, wr = 0;
int xp,x1,x2,x3,yp,y1,y2,y3,theta1,theta2,v;

void setup()
{
  sei();
  EIMSK|=(1<<INT0)|(1<<INT1);
  EICRA|=(1<<ISC01)|(1<<ISC00)|(1<<ISC11)|(1<<ISC10);
  Serial.flush();
  Serial.begin(115200);
  xbee.setSerial(Serial);
  pinMode (rm1, OUTPUT);
  pinMode (rm2, OUTPUT);
  pinMode (rme, OUTPUT);
  pinMode (lm1, OUTPUT);
  pinMode (lm2, OUTPUT);
  pinMode (lme, OUTPUT);
  pinMode (2,INPUT);
  pinMode (3,INPUT);
}

void lm()
{
  lpos=1;
  rpos=0;
  analogWrite(lme,lsp);
  digitalWrite(lm1,HIGH);
  digitalWrite(lm2,LOW);
  digitalWrite(rm1,HIGH);
  digitalWrite(rm2,HIGH);
}

void rm()
{
  rpos=1;
  lpos=0;
  analogWrite(rme,rsp);
  digitalWrite(rm1,HIGH);
  digitalWrite(rm2,LOW);
  digitalWrite(lm1,HIGH);
  digitalWrite(lm2,HIGH);
}

void forward()
{
  lpos=rpos=1;
  analogWrite(lme,lsp);
  digitalWrite(lm1,HIGH);
  digitalWrite(lm2,LOW);
  analogWrite(rme,rsp);
  digitalWrite(rm1,HIGH);
  digitalWrite(rm2,LOW);
}

void reverse()
{
  lpos=rpos=2;
  analogWrite(lme,lsp);
  digitalWrite(lm2,HIGH);
  digitalWrite(lm1,LOW);
  analogWrite(rme,rsp);
  digitalWrite(rm2,HIGH);
  digitalWrite(rm1,LOW);
}

void leftturn()
{
  lpos=2;
  rpos=1;
  analogWrite(lme,lsp);
  digitalWrite(lm2,HIGH);
  digitalWrite(lm1,LOW);
  analogWrite(rme,rsp);
  digitalWrite(rm1,HIGH);
  digitalWrite(rm2,LOW);
}

void rightturn()
{
  rpos=2;
  lpos=1;
  analogWrite(lme,lsp);
  digitalWrite(lm1,HIGH);
  digitalWrite(lm2,LOW);
  analogWrite(rme,rsp);
  digitalWrite(rm2,HIGH);
  digitalWrite(rm1,LOW);
}

void brake()
{
  lpos=rpos=0;
  digitalWrite(lm1,HIGH);
  digitalWrite(lm2,HIGH);
  digitalWrite(rm2,HIGH);
  digitalWrite(rm1,HIGH);
}

void halt()
{
  lpos=rpos=0;
  digitalWrite(lme,LOW);
  digitalWrite(rme,LOW);
}

void track()
{
  eleft=analogRead(0)>400?1:0;
  left=analogRead(1)>400?1:0;
  centre=analogRead(2)>400?1:0;
  right=analogRead(3)>400?1:0;
  eright=analogRead(4)>400?1:0;
  line=eleft*10000+left*1000+centre*100+right*10+eright;

 A:
  if((line==11011)||(line==10001))
    {
      forward();
      pre=line;
    }
  else if((line==10111)||(line==10011))
    {
      lm();
      pre=line;
    }
  else if((line==11101)||(line==11001))
    {
      rm();
      pre=line;
    }
  else if((line==01111)||(line==00111)||(line==00011))
    {
      rightturn();
      pre=line;
    }
  else if((line==11110)||(line==11100)||(line==11000))
    {
      leftturn();
      pre=line;
    }
  else if(line==11111)
    {
      line=pre;
      goto A;
    }
  else if(line==00000)
    {
      brake();
    }
}

unsigned long getElapsedTime(unsigned long prevTime)
{
  unsigned long currentTime = micros();
  if (currentTime < prevTime)
    return UINT_MAX - prevTime + currentTime;
  return currentTime - prevTime;
}

void computeAngularVelocities()
{
  unsigned long dt_omega = getElapsedTime(prevWheelComputeTime);
  prevWheelComputeTime = micros();

  float c = (2 * PI) / ((TICKSPERREV * dt_omega) / 1000000.0);

  wl = (leftTicks - leftTicksPrev) * c;
  wr = (rightTicks - rightTicksPrev) * c;

  leftTicksPrev = leftTicks;
  rightTicksPrev = rightTicks;
}

void computePosition()
{
  computeAngularVelocities();
  unsigned long dt_integration = getElapsedTime(prevIntegrationTime);
  prevIntegrationTime = micros();

  float dt = dt_integration / 1000000.0;
  float Vl = wl * RADIUS;
  float Vr = wr * RADIUS;
  float V = (Vr + Vl) / 2.0;
  float w = (Vr - Vl) / LENGTH;

  v=V;
  float thetaNext = theta + dt * w;
  float xNext = xc + (dt * v*cos(thetaNext))/10.0;
  float yNext = yc + (dt * v*sin(thetaNext))/10.0;

  float distance = sqrt(xc * xc + yc * yc);

  xc = xNext;
  yc = yNext;
  theta = thetaNext;

  if(theta>2*PI)
    theta=(theta-(2*PI));
  else if(theta<0)
    theta=(theta+(2*PI));

  if(xc<0)
  {
    x1=(int(-1*xc))%100;
    x2=(int(-1*xc/100))%100;
    x3=(int(-1*xc/10000))%100;
    xp=1;
  }
  else
  {
    x1=(int(xc))%100;
    x2=(int(xc/100))%100;
    x3=(int(xc/10000))%100;
    xp=0;
  }

  if(yc<0)
  {
    y1=(int(-1*yc))%100;
    y2=(int(-1*yc/100))%100;
    y3=(int(-1*yc/10000))%100;
    yp=1;
  }
  else
  {
    y1=(int(yc))%100;
    y2=(int(yc/100))%100;
    y3=(int(yc/10000))%100;
    yp=0;
  }

  theta1=(int(theta*100))%100;
  theta2=(int(theta*100))/100;
}

void loop()
{
  uint8_t payload[] = {lrpm,rrpm,lpos,rpos,xp,x3,x2,x1,yp,y3,y2,y1,theta2,theta1,v};
  track();
  lrot=lctr/30.0;
  rrot=rctr/30.0;

  if (millis() - prevPositionComputeTime > 50)
  {
    computePosition();
    prevPositionComputeTime = millis();
  }

  if( millis() - start > 100)
  {
    t=millis()-start;
    start=millis();
    lrpm=((lpulse/30.0)*60.0*1000.0)/t;
    rrpm=((rpulse/30.0)*60.0*1000.0)/t;
    lpulse=rpulse=0;
    ZBTxRequest zbTx = ZBTxRequest(addr64_Follower, payload, sizeof(payload));
    xbee.send(zbTx);
    Serial.println("Sending ");
  }
}

ISR(INT1_vect)
{
  lctr++;
  lpulse++;
  if(lpos==2)
    leftTicks--;
  else
    leftTicks++;
  delay(10);
}

ISR(INT0_vect)
{
  rctr++;
  rpulse++;
  if(rpos==2)
    rightTicks--;
  else
```

```
    rightTicks++;
  delay(10);
}
```

5.1.2 Follower

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include <XBee.h>

XBee xbee = XBee();
ZBTxStatusResponse txStatus = ZBTxStatusResponse();
XBeeResponse response = XBeeResponse();
ZBRxResponse rx = ZBRxResponse();

XBeeAddress64 addr64_Leader  = XBeeAddress64(0x00000000, 0x0000FFFF);
XBeeAddress64 addr64_Follower  = XBeeAddress64(0x0013A200, 0x41517A82);
XBeeAddress64 addr64_Laptop  = XBeeAddress64(0x0013A200, 0x415177B1);

#define RADIUS 25.75
#define LENGTH 172
#define TICKSPERREV 30

#define UINT_MAX 4294967295

int rm1=8, rm2=9, rme=10, lm1=12, lm2=13, lme=11;

int setlpos=0,setlrpm=0,setrpos=0,setrrpm=0;
int ldif=0,lid=0,ldd=0,lprev=0,rdif=0,rid=0,rdd=0,rprev=0;
int lsp=0,rsp=0;
int lpre_ocr=0,rpre_ocr=0;
long line=0,pre=0,start=0;
volatile int lpos=0,rpos=0;
float rrot=0,lrot=0;
volatile unsigned long lctr = 0, rctr = 0, lpulse = 0, rpulse = 0, leftTicks = 0, rightTicks = 0, leftTicksPrev = 0, rightTicksPrev = 0;
volatile unsigned long lrpm, rrpm, duration, prev = 0, cur = 0, t = 0;

unsigned long prevPositionComputeTime = 0, prevWheelComputeTime = 0, prevIntegrationTime = 0;
double prevX = 0, prevY = 0, xc = 0, yc = 0, theta = 0, wl = 0, wr = 0;
int sxp=0,syp=0,sx1=0,sy1=0,stheta1=0,sx2=0,sy2=0,stheta2=0,sx3=0,sy3=0;
double sx=0,sy=0,stheta=0,sv;
int xp=0,yp=0,x1=0,y1=0,theta1=0,x2=0,y2=0,theta2=0,x3=0,y3=0;

void setup()
{
  sei();
  EIMSK|=(1<<INT0)|(1<<INT1);
  EICRA|=(1<<ISC01)|(1<<ISC00)|(1<<ISC11)|(1<<ISC10);
  Serial.flush();
  Serial.begin(115200);
  xbee.setSerial(Serial);
  pinMode (rm1, OUTPUT);
  pinMode (rm2, OUTPUT);
  pinMode (rme, OUTPUT);
  pinMode (lm1, OUTPUT);
  pinMode (lm2, OUTPUT);
  pinMode (lme, OUTPUT);
  pinMode (2,INPUT);
  pinMode (3,INPUT);
}

void lmf()
{
  lpos=1;
  rpos=0;
  analogWrite(lme,lsp);
  digitalWrite(lm1,HIGH);
  digitalWrite(lm2,LOW);
}

void rmf()
{
  rpos=1;
  lpos=0;
  analogWrite(rme,rsp);
  digitalWrite(rm1,HIGH);
  digitalWrite(rm2,LOW);
}

void lmb()
{
  lpos=2;
  analogWrite(lme,lsp);
  digitalWrite(lm1,LOW);
  digitalWrite(lm2,HIGH);
}

void rmb()
{
  rpos=2;
  analogWrite(rme,rsp);
  digitalWrite(rm1,LOW);
  digitalWrite(rm2,HIGH);
}

void lmh()
{
  lpos=0;
  analogWrite(lme,lsp);
  digitalWrite(lm1,LOW);
  digitalWrite(lm2,LOW);
}

void rmh()
{
  rpos=0;
  analogWrite(rme,rsp);
  digitalWrite(rm1,LOW);
  digitalWrite(rm2,LOW);
}

unsigned long getElapsedTime(unsigned long prevTime)
{
  unsigned long currentTime = micros();
  if (currentTime < prevTime)
    return UINT_MAX - prevTime + currentTime;
  return currentTime - prevTime;
}

void computeAngularVelocities()
{
  unsigned long dt_omega = getElapsedTime(prevWheelComputeTime);
  prevWheelComputeTime = micros();

  float c = (2 * PI) / ((TICKSPERREV * dt_omega) / 1000000.0);

  wl = (leftTicks - leftTicksPrev) * c;
  wr = (rightTicks - rightTicksPrev) * c;

  leftTicksPrev = leftTicks;
  rightTicksPrev = rightTicks;
}

void computePosition()
{
  computeAngularVelocities();
  unsigned long dt_integration = getElapsedTime(prevIntegrationTime);
  prevIntegrationTime = micros();

  float dt = dt_integration / 1000000.0;
  float Vl = wl * RADIUS;
  float Vr = wr * RADIUS;
  float v = (Vr + Vl) / 2.0;
  float w = (Vr - Vl) / LENGTH;

  float thetaNext = theta + dt * w;
  float xNext = xc + (dt * v*cos(thetaNext))/10.0;
  float yNext = yc + (dt * v*sin(thetaNext))/10.0;

  float distance = sqrt(xc * xc + yc * yc);

  xc = xNext;
  yc = yNext;
  theta = thetaNext;

  if(theta>2*PI)
    theta=(theta-(2*PI));
  else if(theta<0)
    theta=(theta+(2*PI));

  if(xc<0)
  {
    x1=(int(-1*xc))%100;
    x2=(int(-1*xc/100))%100;
    x3=(int(-1*xc/10000))%100;
    xp=1;
  }
  else
  {
    x1=(int(xc))%100;
    x2=(int(xc/100))%100;
    x3=(int(xc/10000))%100;
    xp=0;
  }

  if(yc<0)
  {
    y1=(int(-1*yc))%100;
    y2=(int(-1*yc/100))%100;
    y3=(int(-1*yc/10000))%100;
    yp=1;
  }
  else
  {
    y1=(int(yc))%100;
    y2=(int(yc/100))%100;
    y3=(int(yc/10000))%100;
    yp=0;
  }

  theta1=(int(theta*100))%100;
  theta2=(int(theta*100))/100;
}

void lpid()
{
  ldif=setlrpm-lrpm;
  //lid+=ldif;
  //ldd=ldif-lprev;
  //lprev=ldif;
  lsp=lpre_ocr+((255.0/400.0)*(ldif));
  lpre_ocr=lsp;
  if(lsp>255)
    lsp=255;
}

void rpid()
{
  rdif=setrrpm-rrpm;
  //rid+=rdif;
  //rdd=rdif-rprev;
  //rprev=rdif;
  rsp=rpre_ocr+((255.0/400.0)*(rdif));
  rpre_ocr=rsp;
  if(rsp>255)
    rsp=255;
}

void loop()
{
  lrot=lctr/30.0;
  rrot=rctr/30.0;

  xbee.readPacket();

  if (xbee.getResponse().isAvailable())
  if(xbee.getResponse().getApiId()==ZB_RX_RESPONSE)
  {
    Serial.print("receiving ");
    xbee.getResponse().getZBRxResponse(rx);
    setlrpm=rx.getData(0);
    setrrpm=rx.getData(1);
    setlpos=rx.getData(2);
    setrpos=rx.getData(3);
    sxp=rx.getData(4);
    sx3=rx.getData(5);
    sx2=rx.getData(6);
    sx1=rx.getData(7);
    syp=rx.getData(8);
    sy3=rx.getData(9);
    sy2=rx.getData(10);
    sy1=rx.getData(11);
    stheta2=rx.getData(12);
    stheta1=rx.getData(13);
    sv=rx.getData(14);

    if(sxp==0)
    {
      sx=sx3*10000+sx2*100+sx1;
    }
    else
    {
      sx=-1*(sx3*10000+sx2*100+sx1);
    }
    if(syp==0)
    {
      sy=sy3*10000+sy2*100+sy1;
    }
    else
    {
      sy=-1*(sy3*10000+sy2*100+sy1);
    }
    stheta=(stheta2*100.0+stheta1)/100.0;

    Serial.print(setlrpm);Serial.print(" ");
    Serial.print(setrrpm);Serial.print(" ");
    Serial.print(lrpm);Serial.print(" ");
    Serial.print(rrpm);Serial.print(" ");
    Serial.print(setlpos);Serial.print(" ");
    Serial.println(setrpos);Serial.print(" ");

    if(setlpos==0)
    {
      if(setrpos==0)
      {
        lmh();
        rmh();
      }
      else if(setrpos==1)
      {
        lmh();
        rmf();
      }
      else if(setrpos==2)
      {
        lmh();
        rmb();
      }
    }
    else if(setlpos==1)
    {
      if(setrpos==0)
      {
        lmf();
        rmh();
      }
      else if(setrpos==1)
      {
        lmf();
        rmf();
      }
      else if(setrpos==2)
      {
        lmb();
        rmb();
      }
    }
    else if(setlpos==2)
    {
      if(setrpos==0)
      {
        lmb();
        rmh();
      }
      else if(setrpos==1)
      {
        lmb();
        rmf();
      }
      else if(setrpos==2)
      {
        lmb();
        rmb();
      }
    }
    lpid();
    rpid();

```

```
    uint8_t payload[]={sxp,sx3,sx2,sx1,syp,sy3,sy2,sy1,stheta2,stheta1,xp,x3,x2,x1,yp,y3,y2,y1,theta2,theta1};
    ZBTxRequest zbTx = ZBTxRequest(addr64_Laptop, payload, sizeof(payload));
    xbee.send(zbTx);
  }
  lpid();
  rpid();

  if (millis() - prevPositionComputeTime > 50)
  {
    computePosition();
    prevPositionComputeTime = millis();
  }

  if( millis() - start > 120)
  {
    t=millis()-start;
    start=millis();
    lrpm=((lpulse/30.0)*60.0*1000.0)/t;
    rrpm=((rpulse/30.0)*60.0*1000.0)/t;
    lpulse=rpulse=0;
  }
}

ISR(INT1_vect)
{
  lctr++;
  lpulse++;
  if(lpos==2)
    leftTicks--;
  else
    leftTicks++;
  delay(10);
}

ISR(INT0_vect)
{
  rctr++;
  rpulse++;
  if(rpos==2)
    rightTicks--;
  else
    rightTicks++;
  delay(10);
}
```

5.1.3 Laptop

```
#include <XBee.h>

XBee xbee = XBee();
XBeeResponse response = XBeeResponse();
ZBRxResponse rx = ZBRxResponse();

int sxp=0,syp=0,sx1=0,sy1=0,stheta1=0,sx2=0,sy2=0,stheta2=0,sx3=0,sy3=0;
double sx=0,sy=0,stheta=0;
int xp=0,yp=0,x1=0,y1=0,theta1=0,x2=0,y2=0,theta2=0,x3=0,y3=0;
double x=0,y=0,theta=0;

void setup()
{
 Serial.begin(115200);
 Serial.flush();
 xbee.setSerial(Serial);
}

```

```
void loop()
{
  xbee.readPacket();
  if (xbee.getResponse().isAvailable())
  {
    if(xbee.getResponse().getApiId()==ZB_RX_RESPONSE)
    {
      xbee.getResponse().getZBRxResponse(rx);
      sxp=rx.getData(0);
      sx3=rx.getData(1);
      sx2=rx.getData(2);
      sx1=rx.getData(3);
      syp=rx.getData(4);
      sy3=rx.getData(5);
      sy2=rx.getData(6);
      sy1=rx.getData(7);
      stheta2=rx.getData(8);
      stheta1=rx.getData(9);
      xp=rx.getData(10);
      x3=rx.getData(11);
      x2=rx.getData(12);
      x1=rx.getData(13);
      yp=rx.getData(14);
      y3=rx.getData(15);
      y2=rx.getData(16);
      y1=rx.getData(17);
      theta2=rx.getData(18);
      theta1=rx.getData(19);

      if(sxp==0)
      {
        sx=sx3*10000+sx2*100+sx1;
      }
      else
      {
        sx=-1*(sx3*10000+sx2*100+sx1);
      }
      if(syp==0)
      {
        sy=sy3*10000+sy2*100+sy1;
      }
      else
      {
        sy=-1*(sy3*10000+sy2*100+sy1);
      }
      stheta=(stheta2*100.0+stheta1)/100.0;

      if(xp==0)
      {
        x=x3*10000+x2*100+x1;
      }else
      {
        x=-1*(x3*10000+x2*100+x1);
      }
      if(yp==0)
      {
        y=y3*10000+y2*100+y1;
      }
      else
      {
        y=-1*(y3*10000+y2*100+y1);
      }
      theta=(theta2*100.0+theta1)/100.0;

      Serial.print(sx);Serial.print(" ");
      Serial.print(sy);Serial.print(" ");
      Serial.print(stheta);Serial.print(" ");
      Serial.print(x);Serial.print(" ");
      Serial.print(y);Serial.print(" ");
      Serial.println(theta);
    }
  }
}
```

5.1.4 Python Terminal Path Plotting

```
import serial
import numpy as np
import matplotlib.pyplot as plt
from drawnow import *

ard = serial.Serial('com6',115200)
plt.ion()

x1=[]
y1=[]
theta1=[]
x2=[]
y2=[]
theta2=[]
ctr=0

def plot():
    #plt.ylim(-10,10)
    plt.title('Live Path Tracking')
    plt.grid(True)
    plt.ylabel('Y Coordinate (in cm)')
    plt.xlabel('X Coordinate (in cm)')
    plt.plot(x1,y1,'b-',label='Leader Path')
    plt.plot(x2,y2,'o-',label='Follower Path')
    plt.legend(loc='upper right')
    """
    plt.ticklabel_format(useOffset=False)
    plt2=plt.twinx()
    plt.ylim(-10,10)
    plt.plot(y,'r-',label='Follower Path')
    plt2.set_ylabel('Y Coordinate')
    plt2.ticklabel_format(useOffset=False)
    plt2.legend(loc='upper right')
    """

while True:
    while (ard.inWaiting()==0):
        pass
    string = ard.readline()
    arr = string.split(' ')
    xc1 = float( arr[0] )
    yc1 = float( arr[1] )
    thetac1 = float( arr[2] )
    xc2 = float( arr[3] )
    yc2 = float( arr[4] )
    thetac2 = float( arr[5] )
    x1.append(xc1)
    y1.append(yc1)
    theta1.append(thetac1)
    x2.append(xc2)
    y2.append(yc2)
    theta2.append(thetac2)
    drawnow(plot)
    ctr=ctr+1
    if(ctr>200):
        x1.pop(0)
        y1.pop(0)
        theta1.pop(0)
        x2.pop(0)
        y2.pop(0)
        theta2.pop(0)
```

6 Advanced Task

In this task, a path is hard coded into the leader robot and it moves in that path as a differential drive robot. Simultaneously, it calculates the odometric values i.e, it's x,y coordinates and orientation. All the above data through XBee.

The follower gets the location of the leader and uses kinematics to find the direction and distance of the leader and then moves to the location. It then sends the x,y,theta of both the leader and the follower bots to the Laptop Xbee. The paths are then plotted in Python Serial.

6.1 Source Code

6.1.1 Leader

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include <XBee.h>

XBee xbee = XBee();
ZBTxStatusResponse txStatus = ZBTxStatusResponse();

#define RADIUS 25.75
#define LENGTH 166
#define TICKSPERREV 30

#define UINT_MAX 4294967295

XBeeAddress64 addr64_Broadcast  = XBeeAddress64(0x00000000, 0x0000FFFF);
XBeeAddress64 addr64_Follower  = XBeeAddress64(0x0013A200, 0x41517A82);
XBeeAddress64 addr64_Laptop  = XBeeAddress64(0x0013A200, 0x415177B1);

int eleft,left,centre,right,eright;
int rm1=8, rm2=9, rme=10, lm1=12, lm2=13, lme=11;

float dist=0;
int reached=0;

int lsp=155,rsp=155;
long line=0,pre=0,start=0;
volatile int lpos=0,rpos=0;
float rrot=0,lrot=0;
volatile unsigned long lctr = 0, rctr = 0, lpulse = 0, rpulse = 0;
long leftTicks = 0, rightTicks = 0, leftTicksPrev = 0, rightTicksPrev = 0;
volatile unsigned long lrpm, rrpm, duration, prev = 0, cur = 0, t = 0;

unsigned long prevPositionComputeTime = 0, prevWheelComputeTime = 0, prevIntegrationTime = 0;
double prevX = 0, prevY = 0, xc = 0, yc = 0, theta = 0, wl = 0, wr = 0;
int xp=0,x1=0,x2=0,x3=0,yp=0,y1=0,y2=0,y3=0,theta1=0,theta2=0;

uint8_t payload[] = {xp,x3,x2,x1,yp,y3,y2,y1,theta2,theta1};
ZBTxRequest zbTx = ZBTxRequest(addr64_Follower, payload, sizeof(payload));

void setup()
{
  sei();
  EIMSK|=(1<<INT0)|(1<<INT1);
  EICRA|=(1<<ISC01)|(1<<ISC00)|(1<<ISC11)|(1<<ISC10);
  Serial.flush();
  Serial.begin(115200);
  xbee.setSerial(Serial);
  pinMode (rm1, OUTPUT);
  pinMode (rm2, OUTPUT);
  pinMode (rme, OUTPUT);
  pinMode (lm1, OUTPUT);
  pinMode (lm2, OUTPUT);
  pinMode (lme, OUTPUT);
  pinMode (2,INPUT);
  pinMode (3,INPUT);
}

void lm()
{
  lpos=1;
  rpos=0;
  analogWrite(lme,lsp);
  digitalWrite(lm1,HIGH);
  digitalWrite(lm2,LOW);
  digitalWrite(rm1,HIGH);
  digitalWrite(rm2,HIGH);
}

void rm()
{
  rpos=1;
  lpos=0;
  analogWrite(rme,rsp);
  digitalWrite(rm1,HIGH);
  digitalWrite(rm2,LOW);
  digitalWrite(lm1,HIGH);
  digitalWrite(lm2,HIGH);
}

void forward()
{
  lpos=rpos=1;
  analogWrite(lme,lsp);
  digitalWrite(lm1,HIGH);
  digitalWrite(lm2,LOW);
  analogWrite(rme,rsp);
  digitalWrite(rm1,HIGH);
  digitalWrite(rm2,LOW);
}

void reverse()
{
  lpos=rpos=2;
  analogWrite(lme,lsp);
  digitalWrite(lm2,HIGH);
  digitalWrite(lm1,LOW);
  analogWrite(rme,rsp);
  digitalWrite(rm2,HIGH);
  digitalWrite(rm1,LOW);
}

void leftturn()
{
  lpos=2;
  rpos=1;
  analogWrite(lme,80);
  digitalWrite(lm2,HIGH);
  digitalWrite(lm1,LOW);
  analogWrite(rme,80);
  digitalWrite(rm1,HIGH);
  digitalWrite(rm2,LOW);
}

void rightturn()
{
  rpos=2;
  lpos=1;
  analogWrite(lme,80);
  digitalWrite(lm1,HIGH);
  digitalWrite(lm2,LOW);
  analogWrite(rme,80);
  digitalWrite(rm2,HIGH);
  digitalWrite(rm1,LOW);
}

void halt()
{
  lpos=rpos=0;
  digitalWrite(lm1,LOW);
  digitalWrite(lm2,LOW);
  digitalWrite(rm2,LOW);
  digitalWrite(rm1,LOW);
}

void orient(float st)
{
  if((st-theta)>0.1)
    {
      if((st-theta)<=PI)
      {
        while((theta<(st-0.1))||(theta>(st+0.1)))
          {
            leftturn();
            if (millis() - pre > 50)
            {
              computePosition();
              pre = millis();
            }
            Serial.print(st);Serial.print(" ");
            Serial.println(theta);
          }
          halt();
      }
      else if((st-theta)>PI)
      {
        while((theta<(st-0.1))||(theta>(st+0.1)))
          {
            rightturn();
            if (millis() - pre > 50)
            {
              computePosition();
              pre = millis();
            }
            Serial.print(st);Serial.print(" ");
            Serial.println(theta);
          }
          halt();
      }
    }

    else if((st-theta)<(-0.1))
    {
      if((st-theta)>=PI)
      {
        while((theta<(st-0.1))||(theta>(st+0.1)))
          {
            leftturn();
            if (millis() - pre > 50)
            {
              computePosition();
              pre = millis();
            }
            Serial.print(st);Serial.print(" ");
            Serial.println(theta);
          }
          halt();
      }
      else if((st-theta)<PI)
      {
        while((theta<(st-0.1))||(theta>(st+0.1)))
          {
            rightturn();
            if (millis() - pre > 50)
            {
              computePosition();
              pre = millis();
            }
            Serial.print(st);Serial.print(" ");
            Serial.println(theta);
          }
          halt();
      }
    }
    reached=1;
}

float setang(float sx,float sy)
{
  float ang;
  if(((sy-yc)>=0)&&((sx-xc)>=0))
  {
    ang=atan((sy-yc)/(sx-xc));
  }
  else if(((sy-yc)>=0)&&((sx-xc)<=0))
  {
    ang=PI+atan((sy-yc)/(sx-xc));
  }
  else if(((sy-yc)<=0)&&((sx-xc)<=0))
  {
    ang=PI+atan((sy-yc)/(sx-xc));
  }
  else if(((sy-yc)<=0)&&((sx-xc)>=0))
  {
    ang=2*PI+atan((sy-yc)/(sx-xc));
  }
  return ang;
}

long preTime=0,sendTime=0;

void reach(float sx,float sy,float st=100)
{
  computePosition();
  float inix=xc,iniy=yc;
  orient(setang(sx,sy));
  dist=sqrt(((sy-yc)*(sy-yc))+((sx-xc)*(sx-xc)));
  float d=0.0;

  while(d<(dist))
  {
    if (millis() - preTime > 50)
    {
      computePosition();
      preTime = millis();
    }
    if (millis() - sendTime > 250)
    {
      payload_update();
      xbee.send(zbTx);
      sendTime = millis();
    }
    d=sqrt((yc-iniy)*(yc-iniy)+(xc-inix)*(xc-inix));
    forward();
    if(d>=(dist))
    {
      halt();
      break;
    }
    Serial.print(dist);Serial.print(" ");
    Serial.println(d);
  }
  halt();
  if(st!=100)
    orient(st);
  reached=1;
}

unsigned long getElapsedTime(unsigned long prevTime)
{
  unsigned long currentTime = micros();
  if (currentTime < prevTime)
    return UINT_MAX - prevTime + currentTime;
  return currentTime - prevTime;
}

void computeAngularVelocities()
{
  unsigned long dt_omega = getElapsedTime(prevWheelComputeTime);
  prevWheelComputeTime = micros();

  float c = (2 * PI) / ((TICKSPERREV * dt_omega) / 1000000.0);

  wl = (leftTicks - leftTicksPrev) * c;
  wr = (rightTicks - rightTicksPrev) * c;

  leftTicksPrev = leftTicks;
  rightTicksPrev = rightTicks;
}

void computePosition()
{
  computeAngularVelocities();
  unsigned long dt_integration = getElapsedTime(prevIntegrationTime);
  prevIntegrationTime = micros();

  float dt = dt_integration / 1000000.0;
  float Vl = wl * RADIUS;
  float Vr = wr * RADIUS;
  float v = (Vr + Vl) / 2.0;
  float w = (Vr - Vl) / LENGTH;

  float thetaNext = theta + dt * w;
  float xNext = xc + (dt * v*cos(thetaNext))/10.0;
  float yNext = yc + (dt * v*sin(thetaNext))/10.0;

  float distance = sqrt(xc * xc + yc * yc);

  xc = xNext;
  yc = yNext;
  theta = thetaNext;

  if(theta>2*PI)
    theta=(theta-(2*PI));
  else if(theta<0)
    theta=(theta+(2*PI));

  if(xc<0)
  {
    x1=(int(-1*xc))%100;
    x2=(int(-1*xc/100))%100;
    x3=(int(-1*xc/10000))%100;
    xp=1;
  }
  else
  {
    x1=(int(xc))%100;
    x2=(int(xc/100))%100;
    x3=(int(xc/10000))%100;
    xp=0;
  }

  if(yc<0)
  {
    y1=(int(-1*yc))%100;
    y2=(int(-1*yc/100))%100;
    y3=(int(-1*yc/10000))%100;
    yp=1;
  }
  else
  {
    y1=(int(yc))%100;
    y2=(int(yc/100))%100;
    y3=(int(yc/10000))%100;
    yp=0;
  }

  theta1=(int(theta*100))%100;
  theta2=(int(theta*100))/100;
}

void payload_update()
{
  payload[0] = xp;
  payload[1] = x3;
  payload[2] = x2;
  payload[3] = x1;
  payload[4] = yp;
  payload[5] = y3;
  payload[6] = y2;
  payload[7] = y1;
  payload[8] = theta2;
  payload[9] = theta1;
}

void loop()
{
  payload_update();
  lrot=lctr/30.0;
  rrot=rctr/30.0;
  if(reached==0)
    {
      reached=0;
      payload_update();
      xbee.send(zbTx);
      reach(60.0,14.0);
      payload_update();
      xbee.send(zbTx);
      reach(20.0,22.0);
      payload_update();
      xbee.send(zbTx);
      reach(0.0,60.0);
      payload_update();
      xbee.send(zbTx);
      reach(-20.0,22.0);
      payload_update();
      xbee.send(zbTx);
      reach(-60.0,14.0);
      payload_update();
      xbee.send(zbTx);
      reach(-32.0,-18.0);
      payload_update();
      xbee.send(zbTx);
      reach(-38.0,-60.0);
      payload_update();
      xbee.send(zbTx);
      reach(0.0,-42.0);
      payload_update();
      xbee.send(zbTx);
      reach(38.0,-60.0);
      payload_update();
      xbee.send(zbTx);
      reach(32.0,-18.0);
      payload_update();
      xbee.send(zbTx);
      reach(60.0,14.0);
      payload_update();
      xbee.send(zbTx);
    }

  if(reached==1)
    Serial.println("reached");

  if (millis() - prevPositionComputeTime > 50)
  {
    computePosition();
    prevPositionComputeTime = millis();
  }

  if( millis() - start > 200)
  {
    t=millis()-start;
    start=millis();
    lrpm=((lpulse/30.0)*60.0*1000.0)/t;
    rrpm=((rpulse/30.0)*60.0*1000.0)/t;
    lpulse=rpulse=0;
  }
}

ISR(INT1_vect)
{
  lctr++;
  lpulse++;
  if(lpos==2)
    leftTicks--;
  else
    leftTicks++;
  delay(10);
}

ISR(INT0_vect)
```

```
{
  rctr++;
  rpulse++;
  if(rpos==2)
    rightTicks--;
  else
    rightTicks++;
  delay(10);
}
```

6.1.2 Follower

```
#include <avr/io.h>
#include <avr/interrupt.h>
#include <util/delay.h>
#include <XBee.h>

XBee xbee = XBee();
ZBTxStatusResponse txStatus = ZBTxStatusResponse();
XBeeResponse response = XBeeResponse();
ZBRxResponse rx = ZBRxResponse();

XBeeAddress64 addr64_Leader  = XBeeAddress64(0x00000000, 0x0000FFFF);
XBeeAddress64 addr64_Follower  = XBeeAddress64(0x0013A200, 0x41517A82);
XBeeAddress64 addr64_Laptop  = XBeeAddress64(0x0013A200, 0x415177B1);

#define RADIUS 25.75
#define LENGTH 172
#define TICKSPERREV 30

#define UINT_MAX 4294967295

int rm1=8, rm2=9, rme=10, lm1=12, lm2=13, lme=11;

float dist=0;
int reached=0;

int setlpos=0,setlrpm=0,setrpos=0,setrrpm=0;
int ldif=0,lid=0,ldd=0,lprev=0,rdif=0,rid=0,rdd=0,rprev=0;
int lsp=155,rsp=155;
int lpre_ocr=0,rpre_ocr=0;
long line=0,pre=0,start=0;
volatile int lpos=0,rpos=0;
float rrot=0,lrot=0;
volatile unsigned long lctr = 0, rctr = 0, lpulse = 0, rpulse = 0, leftTicks = 0, rightTicks = 0,
    leftTicksPrev = 0, rightTicksPrev = 0;
volatile unsigned long lrpm, rrpm, duration, prev = 0, cur = 0, t = 0;

unsigned long prevPositionComputeTime = 0, prevWheelComputeTime = 0, prevIntegrationTime = 0;
double prevX = 0, prevY = 0, xc = 0, yc = 0, theta = 0, wl = 0, wr = 0;
int sxp=0,syp=0,sx1=0,sy1=0,stheta1=0,sx2=0,sy2=0,stheta2=0,sx3=0,sy3=0;
double sx=0,sy=0,stheta=0;
int xp=0,yp=0,x1=0,y1=0,theta1=0,x2=0,y2=0,theta2=0,x3=0,y3=0;

uint8_t payload[] = {sxp,sx3,sx2,sx1,syp,sy3,sy2,sy1,stheta2,stheta1,xp,x3,x2,x1,yp,y3,y2,y1,theta2,theta1
    };
ZBTxRequest zbTx = ZBTxRequest(addr64_Laptop, payload, sizeof(payload));

void setup()
{
  sei();
  EIMSK|=(1<<INT0)|(1<<INT1);
  EICRA|=(1<<ISC01)|(1<<ISC00)|(1<<ISC11)|(1<<ISC10);
  Serial.flush();
  Serial.begin(115200);
  xbee.setSerial(Serial);
  pinMode (rm1, OUTPUT);
  pinMode (rm2, OUTPUT);
  pinMode (rme, OUTPUT);
  pinMode (lm1, OUTPUT);
  pinMode (lm2, OUTPUT);
  pinMode (lme, OUTPUT);
  pinMode (2,INPUT);
  pinMode (3,INPUT);
}

void lmf()
{
  lpos=1;
  rpos=0;
  analogWrite(lme,lsp);
  digitalWrite(lm1,HIGH);
  digitalWrite(lm2,LOW);
}

void rmf()
{
  rpos=1;
  lpos=0;
  analogWrite(rme,rsp);
  digitalWrite(rm1,HIGH);
  digitalWrite(rm2,LOW);
}

void lmb()
{
  lpos=2;
  analogWrite(lme,lsp);
  digitalWrite(lm1,LOW);
  digitalWrite(lm2,HIGH);
}

void rmb()
{
  rpos=2;
  analogWrite(rme,rsp);
  digitalWrite(rm1,LOW);
  digitalWrite(rm2,HIGH);
}

void lmh()
{
  lpos=0;
  analogWrite(lme,lsp);
  digitalWrite(lm1,LOW);
  digitalWrite(lm2,LOW);
}

void rmh()
{
  rpos=0;
  analogWrite(rme,rsp);
  digitalWrite(rm1,LOW);
  digitalWrite(rm2,LOW);
}

void leftturn()
{
  lpos=2;
  rpos=1;
  analogWrite(lme,80);
  digitalWrite(lm2,HIGH);
  digitalWrite(lm1,LOW);
  analogWrite(rme,80);
  digitalWrite(rm1,HIGH);
  digitalWrite(rm2,LOW);
}

void righturn()
{
  rpos=2;
  lpos=1;
  analogWrite(lme,80);
  digitalWrite(lm1,HIGH);
  digitalWrite(lm2,LOW);
  analogWrite(rme,80);
  digitalWrite(rm2,HIGH);
  digitalWrite(rm1,LOW);
}

void halt()
{
  lpos=rpos=0;
  digitalWrite(lme,LOW);
  digitalWrite(rme,LOW);
  digitalWrite(lm1,LOW);
  digitalWrite(lm2,LOW);
  digitalWrite(rm2,LOW);
  digitalWrite(rm1,LOW);
}

void forward()
{
  lpos=rpos=1;
  analogWrite(lme,lsp);
  digitalWrite(lm1,HIGH);
  digitalWrite(lm2,LOW);
  analogWrite(rme,rsp);
  digitalWrite(rm1,HIGH);
  digitalWrite(rm2,LOW);
}

void reverse()
{
  lpos=rpos=2;
  analogWrite(lme,lsp);
  digitalWrite(lm2,HIGH);
  digitalWrite(lm1,LOW);
  analogWrite(rme,rsp);
  digitalWrite(rm2,HIGH);
  digitalWrite(rm1,LOW);
}

void orient(float st)
{
  if((st-theta)>0.1)
    {
      if((st-theta)<=PI)
      {
        while((theta<(st-0.1))||(theta>(st+0.1)))
          {
            leftturn();
            if (millis() - pre > 50)
            {
              computePosition();
              pre = millis();
            }
            Serial.print(st);Serial.print(" ");
            Serial.println(theta);
          }
          halt();
      }
      else if((st-theta)>PI)
      {
        while((theta<(st-0.1))||(theta>(st+0.1)))
          {
            rightturn();
            if (millis() - pre > 50)
            {
              computePosition();
              pre = millis();
            }
            Serial.print(st);Serial.print(" ");
            Serial.println(theta);
          }
          halt();
      }
    }

    else if((st-theta)<(-0.1))
    {
      if((st-theta)>=PI)
      {
        while((theta<(st-0.1))||(theta>(st+0.1)))
          {
            leftturn();
            if (millis() - pre > 50)
            {
              computePosition();
              pre = millis();
            }
            Serial.print(st);Serial.print(" ");
            Serial.println(theta);
          }
          halt();
      }
      else if((st-theta)<PI)
      {
        while((theta<(st-0.1))||(theta>(st+0.1)))
          {
            rightturn();
            if (millis() - pre > 50)
            {
              computePosition();
              pre = millis();
            }
            Serial.print(st);Serial.print(" ");
            Serial.println(theta);
          }
          halt();
      }
    }
    reached=1;
}

float setang(float sx,float sy)
{
  float ang;
  if(((sy-yc)>=0)&&((sx-xc)>=0))
  {
    ang=atan((sy-yc)/(sx-xc));
  }
  else if(((sy-yc)>=0)&&((sx-xc)<=0))
  {
    ang=PI+atan((sy-yc)/(sx-xc));
  }
  else if(((sy-yc)<=0)&&((sx-xc)<=0))
  {
    ang=PI+atan((sy-yc)/(sx-xc));
  }
  else if(((sy-yc)<=0)&&((sx-xc)>=0))
  {
    ang=2*PI+atan((sy-yc)/(sx-xc));
  }
  return ang;
}

long preTime=0,sendTime=0;

void reach(float sx,float sy,float st=100)
{
  computePosition();
  float inix=xc,iniy=yc;
  orient(setang(sx,sy));
  dist=sqrt(((sy-yc)*(sy-yc))+((sx-xc)*(sx-xc)));
  float d=0.0;

  while(d<(dist))
  {
    if (millis() - preTime > 50)
    {
      computePosition();
      preTime = millis();
    }
    if (millis() - sendTime > 250)
    {
      payload_update();
      xbee.send(zbTx);
      sendTime = millis();
    }
    d=sqrt((yc-iniy)*(yc-iniy)+(xc-inix)*(xc-inix));
    forward();
    if(d>=(dist))
    {
      halt();
      break;
    }
    Serial.print(dist);Serial.print(" ");
    Serial.println(d);
  }
  halt();
  if(st!=100)
    orient(st);
  reached=1;
}

unsigned long getElapsedTime(unsigned long prevTime)
{
  unsigned long currentTime = micros();
  if (currentTime < prevTime)
    return UINT_MAX - prevTime + currentTime;
  return currentTime - prevTime;
}

void computeAngularVelocities()
{
  unsigned long dt_omega = getElapsedTime(prevWheelComputeTime);
  prevWheelComputeTime = micros();

  float c = (2 * PI) / ((TICKSPERREV * dt_omega) / 1000000.0);

  wl = (leftTicks - leftTicksPrev) * c;
  wr = (rightTicks - rightTicksPrev) * c;

  leftTicksPrev = leftTicks;
  rightTicksPrev = rightTicks;
}

void computePosition()
{
  computeAngularVelocities();
  unsigned long dt_integration = getElapsedTime(prevIntegrationTime);
  prevIntegrationTime = micros();

  float dt = dt_integration / 1000000.0;
  float Vl = wl * RADIUS;
  float Vr = wr * RADIUS;
  float v = (Vr + Vl) / 2.0;
  float w = (Vr - Vl) / LENGTH;

  float thetaNext = theta + dt * w;
  float xNext = xc + (dt * v*cos(thetaNext))/10.0;
  float yNext = yc + (dt * v*sin(thetaNext))/10.0;

  float distance = sqrt(xc * xc + yc * yc);

  xc = xNext;
  yc = yNext;
  theta = thetaNext;

  if(theta>2*PI)
    theta=(theta-(2*PI));
  else if(theta<0)
    theta=(theta+(2*PI));

  if(xc<0)
  {
    x1=(int(-1*xc))%100;
    x2=(int(-1*xc/100))%100;
    x3=(int(-1*xc/10000))%100;
    xp=1;
  }
  else
  {
    x1=(int(xc))%100;
    x2=(int(xc/100))%100;
    x3=(int(xc/10000))%100;
    xp=0;
  }

  if(yc<0)
  {
    y1=(int(-1*yc))%100;
    y2=(int(-1*yc/100))%100;
    y3=(int(-1*yc/10000))%100;
    yp=1;
  }
  else
  {
    y1=(int(yc))%100;
    y2=(int(yc/100))%100;
    y3=(int(yc/10000))%100;
    yp=0;
  }

  theta1=(int(theta*100))%100;
  theta2=(int(theta*100))/100;
}

void lpid()
{
  ldif=setlrpm-lrpm;
  //lid+=ldif;
  //ldd=ldif-lprev;
  //lprev=ldif;
  lsp=lpre_ocr+((255.0/400.0)*(ldif));
  lpre_ocr=lsp;
  if(lsp>255)
    lsp=255;
}

void rpid()
{
  rdif=setrrpm-rrpm;
  //rid+=rdif;
  //rdd=rdif-rprev;
  //rprev=rdif;
  rsp=rpre_ocr+((255.0/400.0)*(rdif));
  rpre_ocr=rsp;
  if(rsp>255)
    rsp=255;
}

void payload_update()
{
  payload[0] = xp;
  payload[1] = x3;
  payload[2] = x2;
  payload[3] = x1;
  payload[4] = yp;
  payload[5] = y3;
  payload[6] = y2;
  payload[7] = y1;
  payload[8] = theta2;
  payload[9] = theta1;
}

void loop()
{
  lrot=lctr/30.0;
  rrot=rctr/30.0;

  xbee.readPacket();

  if (xbee.getResponse().isAvailable())
  if(xbee.getResponse().getApiId()==ZB_RX_RESPONSE)
  {
    xbee.getResponse().getZBRxResponse(rx);
    sxp=rx.getData(0);
    sx3=rx.getData(1);
    sx2=rx.getData(2);
    sx1=rx.getData(3);
    syp=rx.getData(4);
    sy3=rx.getData(5);
    sy2=rx.getData(6);
    sy1=rx.getData(7);
    stheta2=rx.getData(8);
    stheta1=rx.getData(9);

    if(sxp==0)
    {
      sx=sx3*10000+sx2*100+sx1;
    }
    else
    {
      sx=-1*(sx3*10000+sx2*100+sx1);
    }
    if(syp==0)
    {
      sy=sy3*10000+sy2*100+sy1;
    }
    else
    {
      sy=-1*(sy3*10000+sy2*100+sy1);
    }
    stheta=(stheta2*100.0+stheta1)/100.0;

    Serial.print(sx);Serial.print(" ");
    Serial.print(sy);Serial.print(" ");
    Serial.println(stheta);

    reach(sx,sy);
    payload_update();
    xbee.send(zbTx);
  }

  if (millis() - prevPositionComputeTime > 50)
  {
    computePosition();
    prevPositionComputeTime = millis();
  }

  if( millis() - start > 200)
  {
    t=millis()-start;
    start=millis();
    lrpm=((lpulse/30.0)*60.0*1000.0)/t;
    rrpm=((rpulse/30.0)*60.0*1000.0)/t;
    lpulse=rpulse=0;
  }
}

ISR(INT1_vect)
{
  lctr++;
  lpulse++;
  if(lpos==2)
    leftTicks--;
  else
    leftTicks++;
  delay(10);
}

ISR(INT0_vect)
{
  rctr++;
  rpulse++;
  if(rpos==2)
    rightTicks--;
  else
    rightTicks++;
  delay(10);
}
```

6.1.3 Laptop

```
#include <XBee.h>

```

```
XBee xbee = XBee();
XBeeResponse response = XBeeResponse();
ZBRxResponse rx = ZBRxResponse();

int sxp=0,syp=0,sx1=0,sy1=0,stheta1=0,sx2=0,sy2=0,stheta2=0,sx3=0,sy3=0;
double sx=0,sy=0,stheta=0;
int xp=0,yp=0,x1=0,y1=0,theta1=0,x2=0,y2=0,theta2=0,x3=0,y3=0;
double x=0,y=0,theta=0;

void setup()
{
 Serial.begin(115200);
 Serial.flush();
 xbee.setSerial(Serial);
}

void loop()
{
  xbee.readPacket();
  if (xbee.getResponse().isAvailable())
  {
    if(xbee.getResponse().getApiId()==ZB_RX_RESPONSE)
    {
      xbee.getResponse().getZBRxResponse(rx);
      sxp=rx.getData(0);
      sx3=rx.getData(1);
      sx2=rx.getData(2);
      sx1=rx.getData(3);
      syp=rx.getData(4);
      sy3=rx.getData(5);
      sy2=rx.getData(6);
      sy1=rx.getData(7);
      stheta2=rx.getData(8);
      stheta1=rx.getData(9);
      xp=rx.getData(10);
      x3=rx.getData(11);
      x2=rx.getData(12);
      x1=rx.getData(13);
      yp=rx.getData(14);
      y3=rx.getData(15);
      y2=rx.getData(16);
      y1=rx.getData(17);
      theta2=rx.getData(18);
      theta1=rx.getData(19);

      if(sxp==0)
      {
        sx=sx3*10000+sx2*100+sx1;
      }
      else
      {
        sx=-1*(sx3*10000+sx2*100+sx1);
      }
      if(syp==0)
      {
        sy=sy3*10000+sy2*100+sy1;
      }
      else
      {
        sy=-1*(sy3*10000+sy2*100+sy1);
      }
      stheta=(stheta2*100.0+stheta1)/100.0;

      if(xp==0)
      {
        x=x3*10000+x2*100+x1;
      }else
      {
        x=-1*(x3*10000+x2*100+x1);
      }
      if(yp==0)
      {
        y=y3*10000+y2*100+y1;
      }
      else
      {
        y=-1*(y3*10000+y2*100+y1);
      }
      theta=(theta2*100.0+theta1)/100.0;

      Serial.print(sx);Serial.print(" ");
      Serial.print(sy);Serial.print(" ");
      Serial.print(stheta);Serial.print(" ");
      Serial.print(x);Serial.print(" ");
      Serial.print(y);Serial.print(" ");
      Serial.println(theta);
    }
  }
}
```

6.1.4 Python Terminal Path Plotting

```
import serial
import numpy as np
import matplotlib.pyplot as plt
from drawnow import *

ard = serial.Serial('com6',115200)
plt.ion()

x1=[]
y1=[]
theta1=[]
x2=[]
y2=[]
theta2=[]
ctr=0

def plot():
    #plt.ylim(-10,10)
    plt.title('Live Path Tracking')
    plt.grid(True)
    plt.ylabel('Y Coordinate (in cm)')
    plt.xlabel('X Coordinate (in cm)')
    plt.plot(x1,y1,'b-',label='Leader Path')
    plt.plot(x2,y2,'o-',label='Follower Path')
    plt.legend(loc='upper right')
    """
    plt.ticklabel_format(useOffset=False)
    plt2=plt.twinx()
    plt.ylim(-10,10)
    plt.plot(y,'r-',label='Follower Path')
    plt2.set_ylabel('Y Coordinate')
    plt2.ticklabel_format(useOffset=False)
    plt2.legend(loc='upper right')
    """

while True:
    while (ard.inWaiting()==0):
        pass
    string = ard.readline()
    arr = string.split(' ')
    xc1 = float( arr[0] )
    yc1 = float( arr[1] )
    thetac1 = float( arr[2] )
    xc2 = float( arr[3] )
    yc2 = float( arr[4] )
    thetac2 = float( arr[5] )
    x1.append(xc1)
    y1.append(yc1)
    theta1.append(thetac1)
    x2.append(xc2)
    y2.append(yc2)
    theta2.append(thetac2)
    drawnow(plot)
    ctr=ctr+1
    if(ctr>200):
        x1.pop(0)
        y1.pop(0)
        theta1.pop(0)
        x2.pop(0)
        y2.pop(0)
        theta2.pop(0)
```

7 Problems Faced

The main problem I faced is communicating between the XBee devices. There have been a lot of data packet losses at high frequency transmission. I consumed a lot of time in identifying a suitable firmware and configuration. The firmware I have used initially is 802.15.4 which is the firmware of series1 xbee. Zigbee Th reg is the firmware of series2. That's why series 2 code didn't work for 802 firmware. I tried series1 code with 802 firmware, but it failed. I changed configurations in xctu from 802.15.4 to zigbee th reg firmware with series2 code. With this firmware in broadcast, the data is transmitted but the frequency was a problem. Broadcast takes a lot of time as it makes roughly eight transmissions per packet and hence for three XBees make it twenty-four transmissions. So, instead of broadcasting, I used unicast using the MAC addresses. The accuracy of broadcast is one msg in 1.5 sec, whereas in unicast, it is once in 20 ms when tested in xctu. But it doesn't work with same accuracy in arduino sketch in api mode 1, so I changed it to api2. Also the data packet size also determines the time interval between two transmissions. I sent the data from leader to follower in unicast at 20ms delay. From follower I sent the same data as and when received to the laptop in unicast again. Now the problem is solved.

Another problem I faced is that the ir tachometer interrupt bounces too often. To solve this first, I used a comparator circuit, but that too failed. Finally, Schmitt Trigger IC solved the problem.

8 Drive Link for Working Videos

https://drive.google.com/open?id=1zBKGfHVKjvI53KXExJpZJE1W4txeaJRD