NATIONAL INSTITUTE OF TECHNOLOGY
TIRUCHIRAPPALLI

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.

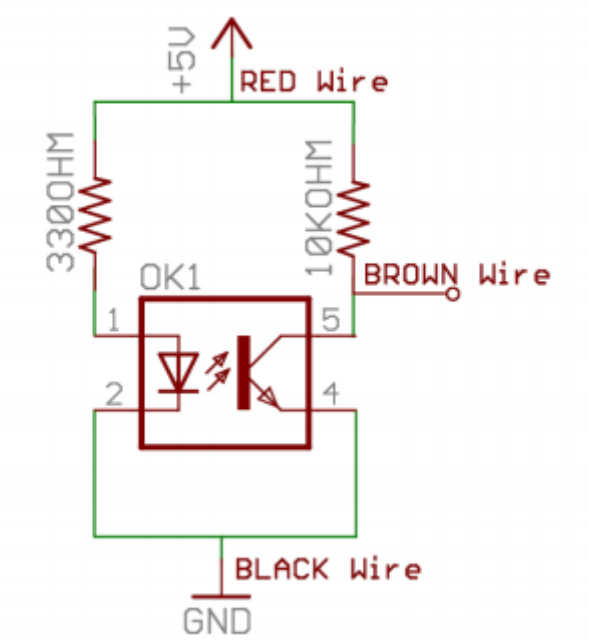
4.1.1 API frame structure

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:

Start delimiter	Length		Frame data								Checksum
			Frame type	Data							
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)



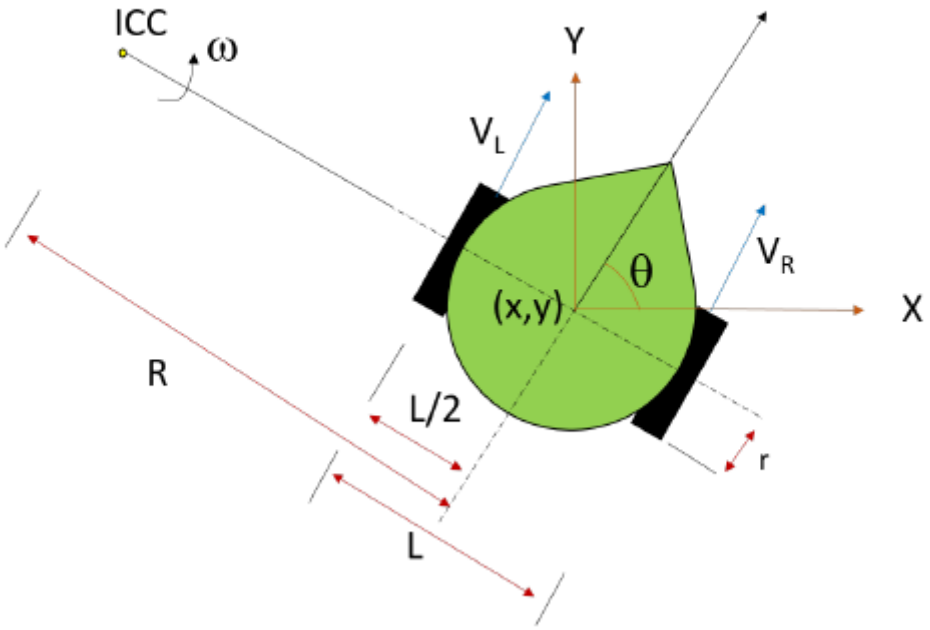
4.3 PID Controller

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);
```

```
46    pinMode (lm2, OUTPUT);
47    pinMode (lme, OUTPUT);
48    pinMode (2,INPUT);
49    pinMode (3,INPUT);
50  }
51
52  void lm()
53  {
54    lpos=1;
55    rpos=0;
56    analogWrite(lme,lsp);
57    digitalWrite(lm1,HIGH);
58    digitalWrite(lm2,LOW);
59    digitalWrite(rm1,HIGH);
60    digitalWrite(rm2,HIGH);
61  }
62
63  void rm()
64  {
65    rpos=1;
66    lpos=0;
67    analogWrite(rme,rsp);
68    digitalWrite(rm1,HIGH);
69    digitalWrite(rm2,LOW);
70    digitalWrite(lm1,HIGH);
71    digitalWrite(lm2,HIGH);
72  }
73
74  void forward()
75  {
76    lpos=rpos=1;
77    analogWrite(lme,lsp);
78    digitalWrite(lm1,HIGH);
79    digitalWrite(lm2,LOW);
80    analogWrite(rme,rsp);
81    digitalWrite(rm1,HIGH);
82    digitalWrite(rm2,LOW);
83  }
84
85  void reverse()
86  {
87    lpos=rpos=2;
88    analogWrite(lme,lsp);
89    digitalWrite(lm2,HIGH);
90    digitalWrite(lm1,LOW);
91    analogWrite(rme,rsp);
92    digitalWrite(rm2,HIGH);
93    digitalWrite(rm1,LOW);
94  }
95
96  void leftturn()
97  {
98    lpos=2;
99    rpos=1;
100   analogWrite(lme,lsp);
101   digitalWrite(lm2,HIGH);
102   digitalWrite(lm1,LOW);
103   analogWrite(rme,rsp);
104   digitalWrite(rm1,HIGH);
105   digitalWrite(rm2,LOW);
106 }
107
108 void rightturn()
109 {
110   rpos=2;
111   lpos=1;
```

```
112   analogWrite(lme,lsp);
113   digitalWrite(lm1,HIGH);
114   digitalWrite(lm2,LOW);
115   analogWrite(rme,rsp);
116   digitalWrite(rm2,HIGH);
117   digitalWrite(rm1,LOW);
118 }
119
120 void brake()
121 {
122   lpos=rpos=0;
123   digitalWrite(lm1,HIGH);
124   digitalWrite(lm2,HIGH);
125   digitalWrite(rm2,HIGH);
126   digitalWrite(rm1,HIGH);
127 }
128
129 void halt()
130 {
131   lpos=rpos=0;
132   digitalWrite(lme,LOW);
133   digitalWrite(rme,LOW);
134 }
135
136 void track()
137 {
138   eleft=analogRead(0)>400?1:0;
139   left=analogRead(1)>400?1:0;
140   centre=analogRead(2)>400?1:0;
141   right=analogRead(3)>400?1:0;
142   eright=analogRead(4)>400?1:0;
143   line=eleft*10000+left*1000+centre*100+right*10+eright;
144
145  A:
146   if((line==11011)||(line==10001))
147     {
148       forward();
149       pre=line;
150     }
151   else if((line==10111)||(line==10011))
152     {
153       lm();
154       pre=line;
155     }
156   else if((line==11101)||(line==11001))
157     {
158       rm();
159       pre=line;
160     }
161   else if((line==01111)||(line==00111)||(line==00011))
162     {
163       rightturn();
164       pre=line;
165     }
166   else if((line==11110)||(line==11100)||(line==11000))
167     {
168       leftturn();
169       pre=line;
170     }
171   else if(line==11111)
172     {
173       line=pre;
174       goto A;
175     }
176   else if(line==00000)
177     {
```

```
178         brake();
179       }
180 }
181
182 unsigned long getElapsedTime(unsigned long prevTime)
183 {
184   unsigned long currentTime = micros();
185   if (currentTime < prevTime)
186     return UINT_MAX - prevTime + currentTime;
187   return currentTime - prevTime;
188 }
189
190 void computeAngularVelocities()
191 {
192   unsigned long dt_omega = getElapsedTime(prevWheelComputeTime);
193   prevWheelComputeTime = micros();
194
195   float c = (2 * PI) / ((TICKSPERREV * dt_omega) / 1000000.0);
196
197   wl = (leftTicks - leftTicksPrev) * c;
198   wr = (rightTicks - rightTicksPrev) * c;
199
200   leftTicksPrev = leftTicks;
201   rightTicksPrev = rightTicks;
202 }
203
204 void computePosition()
205 {
206   computeAngularVelocities();
207   unsigned long dt_integration = getElapsedTime(prevIntegrationTime);
208   prevIntegrationTime = micros();
209
210   float dt = dt_integration / 1000000.0;
211   float Vl = wl * RADIUS;
212   float Vr = wr * RADIUS;
213   float V = (Vr + Vl) / 2.0;
214   float w = (Vr - Vl) / LENGTH;
215
216   v=V;
217   float thetaNext = theta + dt * w;
218   float xNext = xc + (dt * v*cos(thetaNext))/10.0;
219   float yNext = yc + (dt * v*sin(thetaNext))/10.0;
220
221   float distance = sqrt(xc * xc + yc * yc);
222
223   xc = xNext;
224   yc = yNext;
225   theta = thetaNext;
226
227   if(theta>2*PI)
228     theta=(theta-(2*PI));
229   else if(theta<0)
230     theta=(theta+(2*PI));
231
232   if(xc<0)
233   {
234     x1=(int(-1*xc))%100;
235     x2=(int(-1*xc/100))%100;
236     x3=(int(-1*xc/10000))%100;
237     xp=1;
238   }
239   else
240   {
241     x1=(int(xc))%100;
242     x2=(int(xc/100))%100;
243     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;
```

```
126   rightTicksPrev = rightTicks;
127 }
128
129 void computePosition()
130 {
131   computeAngularVelocities();
132   unsigned long dt_integration = getElapsedTime(prevIntegrationTime);
133   prevIntegrationTime = micros();
134
135   float dt = dt_integration / 1000000.0;
136   float Vl = wl * RADIUS;
137   float Vr = wr * RADIUS;
138   float v = (Vr + Vl) / 2.0;
139   float w = (Vr - Vl) / LENGTH;
140
141   float thetaNext = theta + dt * w;
142   float xNext = xc + (dt * v*cos(thetaNext))/10.0;
143   float yNext = yc + (dt * v*sin(thetaNext))/10.0;
144
145   float distance = sqrt(xc * xc + yc * yc);
146
147   xc = xNext;
148   yc = yNext;
149   theta = thetaNext;
150
151   if(theta>2*PI)
152     theta=(theta-(2*PI));
153   else if(theta<0)
154     theta=(theta+(2*PI));
155
156   if(xc<0)
157   {
158     x1=(int(-1*xc))%100;
159     x2=(int(-1*xc/100))%100;
160     x3=(int(-1*xc/10000))%100;
161     xp=1;
162   }
163   else
164   {
165     x1=(int(xc))%100;
166     x2=(int(xc/100))%100;
167     x3=(int(xc/10000))%100;
168     xp=0;
169   }
170
171   if(yc<0)
172   {
173     y1=(int(-1*yc))%100;
174     y2=(int(-1*yc/100))%100;
175     y3=(int(-1*yc/10000))%100;
176     yp=1;
177   }
178   else
179   {
180     y1=(int(yc))%100;
181     y2=(int(yc/100))%100;
182     y3=(int(yc/10000))%100;
183     yp=0;
184   }
185
186   theta1=(int(theta*100))%100;
187   theta2=(int(theta*100))/100;
188 }
189
190 void lpid()
191 {
```

```
192   ldif=setlrpm-lrpm;
193   //lid+=ldif;
194   //ldd=ldif-lprev;
195   //lprev=ldif;
196   lsp=lpre_ocr+((255.0/400.0)*(ldif));
197   lpre_ocr=lsp;
198   if(lsp>255)
199     lsp=255;
200 }
201
202 void rpid()
203 {
204   rdif=setrrpm-rrpm;
205   //rid+=rdif;
206   //rdd=rdif-rprev;
207   //rprev=rdif;
208   rsp=rpre_ocr+((255.0/400.0)*(rdif));
209   rpre_ocr=rsp;
210   if(rsp>255)
211     rsp=255;
212 }
213
214 void loop()
215 {
216   lrot=lctr/30.0;
217   rrot=rctr/30.0;
218
219   xbee.readPacket();
220
221   if (xbee.getResponse().isAvailable())
222   if(xbee.getResponse().getApiId()==ZB_RX_RESPONSE)
223   {
224     Serial.print("receiving ");
225     xbee.getResponse().getZBRxResponse(rx);
226     setlrpm=rx.getData(0);
227     setrrpm=rx.getData(1);
228     setlpos=rx.getData(2);
229     setrpos=rx.getData(3);
230     sxp=rx.getData(4);
231     sx3=rx.getData(5);
232     sx2=rx.getData(6);
233     sxl=rx.getData(7);
234     syp=rx.getData(8);
235     sy3=rx.getData(9);
236     sy2=rx.getData(10);
237     syl=rx.getData(11);
238     stheta2=rx.getData(12);
239     stheta1=rx.getData(13);
240     sv=rx.getData(14);
241
242     if(sxp==0)
243     {
244       sx=sx3*10000+sx2*100+sxl;
245     }
246     else
247     {
248       sx=-1*(sx3*10000+sx2*100+sxl);
249     }
250     if(syp==0)
251     {
252       sy=sy3*10000+sy2*100+syl;
253     }
254     else
255     {
256       sy=-1*(sy3*10000+sy2*100+syl);
257     }
```

```
258     stheta=(stheta2*100.0+stheta1)/100.0;
259
260     Serial.print(setlrpm);Serial.print(" ");
261     Serial.print(setrrpm);Serial.print(" ");
262     Serial.print(lrpm);Serial.print(" ");
263     Serial.print(rrpm);Serial.print(" ");
264     Serial.print(setlpos);Serial.print(" ");
265     Serial.println(setrpos);Serial.print(" ");
266
267     if(setlpos==0)
268     {
269       if(setrpos==0)
270       {
271         lmh();
272         rmh();
273       }
274       else if(setrpos==1)
275       {
276         lmh();
277         rmf();
278       }
279       else if(setrpos==2)
280       {
281         lmh();
282         rmb();
283       }
284     }
285     else if(setlpos==1)
286     {
287       if(setrpos==0)
288       {
289         lmf();
290         rmh();
291       }
292       else if(setrpos==1)
293       {
294         lmf();
295         rmf();
296       }
297       else if(setrpos==2)
298       {
299         lmb();
300         rmb();
301       }
302     }
303     else if(setlpos==2)
304     {
305       if(setrpos==0)
306       {
307         lmb();
308         rmh();
309       }
310       else if(setrpos==1)
311       {
312         lmb();
313         rmf();
314       }
315       else if(setrpos==2)
316       {
317         lmb();
318         rmb();
319       }
320     }
321     lpid();
322     rpid();
323
```

```
    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);
}

```

```
19 void loop()
20 {
21   xbee.readPacket();
22   if (xbee.getResponse().isAvailable())
23   {
24     if(xbee.getResponse().getApiId()==ZB_RX_RESPONSE)
25     {
26       xbee.getResponse().getZBRxResponse(rx);
27       sxp=rx.getData(0);
28       sx3=rx.getData(1);
29       sx2=rx.getData(2);
30       sx1=rx.getData(3);
31       syp=rx.getData(4);
32       sy3=rx.getData(5);
33       sy2=rx.getData(6);
34       sy1=rx.getData(7);
35       stheta2=rx.getData(8);
36       stheta1=rx.getData(9);
37       xp=rx.getData(10);
38       x3=rx.getData(11);
39       x2=rx.getData(12);
40       x1=rx.getData(13);
41       yp=rx.getData(14);
42       y3=rx.getData(15);
43       y2=rx.getData(16);
44       y1=rx.getData(17);
45       theta2=rx.getData(18);
46       theta1=rx.getData(19);
47
48       if(sxp==0)
49       {
50         sx=sx3*10000+sx2*100+sx1;
51       }
52       else
53       {
54         sx=-1*(sx3*10000+sx2*100+sx1);
55       }
56       if(syp==0)
57       {
58         sy=sy3*10000+sy2*100+sy1;
59       }
60       else
61       {
62         sy=-1*(sy3*10000+sy2*100+sy1);
63       }
64       stheta=(stheta2*100.0+stheta1)/100.0;
65
66       if(xp==0)
67       {
68         x=x3*10000+x2*100+x1;
69       }else
70       {
71         x=-1*(x3*10000+x2*100+x1);
72       }
73       if(yp==0)
74       {
75         y=y3*10000+y2*100+y1;
76       }
77       else
78       {
79         y=-1*(y3*10000+y2*100+y1);
80       }
81       theta=(theta2*100.0+theta1)/100.0;
82
83       Serial.print(sx);Serial.print(" ");
84       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);
```

```
45    Serial.flush();
46    Serial.begin(115200);
47    xbee.setSerial(Serial);
48    pinMode (rm1, OUTPUT);
49    pinMode (rm2, OUTPUT);
50    pinMode (rme, OUTPUT);
51    pinMode (lm1, OUTPUT);
52    pinMode (lm2, OUTPUT);
53    pinMode (lme, OUTPUT);
54    pinMode (2,INPUT);
55    pinMode (3,INPUT);
56  }
57
58  void lm()
59  {
60    lpos=1;
61    rpos=0;
62    analogWrite(lme,lsp);
63    digitalWrite(lm1,HIGH);
64    digitalWrite(lm2,LOW);
65    digitalWrite(rm1,HIGH);
66    digitalWrite(rm2,HIGH);
67  }
68
69  void rm()
70  {
71    rpos=1;
72    lpos=0;
73    analogWrite(rme,rsp);
74    digitalWrite(rm1,HIGH);
75    digitalWrite(rm2,LOW);
76    digitalWrite(lm1,HIGH);
77    digitalWrite(lm2,HIGH);
78  }
79
80  void forward()
81  {
82    lpos=rpos=1;
83    analogWrite(lme,lsp);
84    digitalWrite(lm1,HIGH);
85    digitalWrite(lm2,LOW);
86    analogWrite(rme,rsp);
87    digitalWrite(rm1,HIGH);
88    digitalWrite(rm2,LOW);
89  }
90
91  void reverse()
92  {
93    lpos=rpos=2;
94    analogWrite(lme,lsp);
95    digitalWrite(lm2,HIGH);
96    digitalWrite(lm1,LOW);
97    analogWrite(rme,rsp);
98    digitalWrite(rm2,HIGH);
99    digitalWrite(rm1,LOW);
100 }
101
102 void leftturn()
103 {
104   lpos=2;
105   rpos=1;
106   analogWrite(lme,80);
107   digitalWrite(lm2,HIGH);
108   digitalWrite(lm1,LOW);
109   analogWrite(rme,80);
110   digitalWrite(rm1,HIGH);
```

```
111   digitalWrite(rm2,LOW);
112 }
113
114 void rightturn()
115 {
116   rpos=2;
117   lpos=1;
118   analogWrite(lme,80);
119   digitalWrite(lm1,HIGH);
120   digitalWrite(lm2,LOW);
121   analogWrite(rme,80);
122   digitalWrite(rm2,HIGH);
123   digitalWrite(rm1,LOW);
124 }
125
126 void halt()
127 {
128   lpos=rpos=0;
129   digitalWrite(lm1,LOW);
130   digitalWrite(lm2,LOW);
131   digitalWrite(rm2,LOW);
132   digitalWrite(rm1,LOW);
133 }
134
135 void orient(float st)
136 {
137   if((st-theta)>0.1)
138     {
139       if((st-theta)<=PI)
140       {
141         while((theta<(st-0.1))||(theta>(st+0.1)))
142           {
143             leftturn();
144             if (millis() - pre > 50)
145             {
146               computePosition();
147               pre = millis();
148             }
149             Serial.print(st);Serial.print(" ");
150             Serial.println(theta);
151           }
152         halt();
153       }
154       else if((st-theta)>PI)
155       {
156         while((theta<(st-0.1))||(theta>(st+0.1)))
157           {
158             rightturn();
159             if (millis() - pre > 50)
160             {
161               computePosition();
162               pre = millis();
163             }
164             Serial.print(st);Serial.print(" ");
165             Serial.println(theta);
166           }
167           halt();
168       }
169     }
170
171     else if((st-theta)<(-0.1))
172     {
173       if((st-theta)>=PI)
174       {
175         while((theta<(st-0.1))||(theta>(st+0.1)))
176           {
```

```
177             leftturn();
178             if (millis() - pre > 50)
179             {
180               computePosition();
181               pre = millis();
182             }
183             Serial.print(st);Serial.print(" ");
184             Serial.println(theta);
185           }
186           halt();
187       }
188       else if((st-theta)<PI)
189       {
190         while((theta<(st-0.1))||(theta>(st+0.1)))
191           {
192             rightturn();
193             if (millis() - pre > 50)
194             {
195               computePosition();
196               pre = millis();
197             }
198             Serial.print(st);Serial.print(" ");
199             Serial.println(theta);
200           }
201           halt();
202       }
203     }
204     reached=1;
205 }
206
207 float setang(float sx,float sy)
208 {
209   float ang;
210   if(((sy-yc)>=0)&&((sx-xc)>=0))
211   {
212     ang=atan((sy-yc)/(sx-xc));
213   }
214   else if(((sy-yc)>=0)&&((sx-xc)<=0))
215   {
216     ang=PI+atan((sy-yc)/(sx-xc));
217   }
218   else if(((sy-yc)<=0)&&((sx-xc)<=0))
219   {
220     ang=PI+atan((sy-yc)/(sx-xc));
221   }
222   else if(((sy-yc)<=0)&&((sx-xc)>=0))
223   {
224     ang=2*PI+atan((sy-yc)/(sx-xc));
225   }
226   return ang;
227 }
228
229 long preTime=0,sendTime=0;
230
231 void reach(float sx,float sy,float st=100)
232 {
233   computePosition();
234   float inix=xc,iniy=yc;
235   orient(setang(sx,sy));
236   dist=sqrt(((sy-yc)*(sy-yc))+((sx-xc)*(sx-xc)));
237   float d=0.0;
238
239   while(d<(dist))
240   {
241     if (millis() - preTime > 50)
242     {
```

```
      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);
```

```
53    pinMode (rm1, OUTPUT);
54    pinMode (rm2, OUTPUT);
55    pinMode (rme, OUTPUT);
56    pinMode (lm1, OUTPUT);
57    pinMode (lm2, OUTPUT);
58    pinMode (lme, OUTPUT);
59    pinMode (2,INPUT);
60    pinMode (3,INPUT);
61  }
62
63  void lmf()
64  {
65    lpos=1;
66    rpos=0;
67    analogWrite(lme,lsp);
68    digitalWrite(lm1,HIGH);
69    digitalWrite(lm2,LOW);
70  }
71
72  void rmf()
73  {
74    rpos=1;
75    lpos=0;
76    analogWrite(rme,rsp);
77    digitalWrite(rm1,HIGH);
78    digitalWrite(rm2,LOW);
79  }
80
81  void lmb()
82  {
83    lpos=2;
84    analogWrite(lme,lsp);
85    digitalWrite(lm1,LOW);
86    digitalWrite(lm2,HIGH);
87  }
88
89  void rmb()
90  {
91    rpos=2;
92    analogWrite(rme,rsp);
93    digitalWrite(rm1,LOW);
94    digitalWrite(rm2,HIGH);
95  }
96
97  void lmh()
98  {
99    lpos=0;
100   analogWrite(lme,lsp);
101   digitalWrite(lm1,LOW);
102   digitalWrite(lm2,LOW);
103 }
104
105 void rmh()
106 {
107   rpos=0;
108   analogWrite(rme,rsp);
109   digitalWrite(rm1,LOW);
110   digitalWrite(rm2,LOW);
111 }
112
113 void leftturn()
114 {
115   lpos=2;
116   rpos=1;
117   analogWrite(lme,80);
118   digitalWrite(lm2,HIGH);
```

```
119   digitalWrite(lm1,LOW);
120   analogWrite(rme,80);
121   digitalWrite(rm1,HIGH);
122   digitalWrite(rm2,LOW);
123 }
124
125 void rightturn()
126 {
127   rpos=2;
128   lpos=1;
129   analogWrite(lme,80);
130   digitalWrite(lm1,HIGH);
131   digitalWrite(lm2,LOW);
132   analogWrite(rme,80);
133   digitalWrite(rm2,HIGH);
134   digitalWrite(rm1,LOW);
135 }
136
137 void halt()
138 {
139   lpos=rpos=0;
140   digitalWrite(lme,LOW);
141   digitalWrite(rme,LOW);
142   digitalWrite(lm1,LOW);
143   digitalWrite(lm2,LOW);
144   digitalWrite(rm2,LOW);
145   digitalWrite(rm1,LOW);
146 }
147
148 void forward()
149 {
150   lpos=rpos=1;
151   analogWrite(lme,lsp);
152   digitalWrite(lm1,HIGH);
153   digitalWrite(lm2,LOW);
154   analogWrite(rme,rsp);
155   digitalWrite(rm1,HIGH);
156   digitalWrite(rm2,LOW);
157 }
158
159 void reverse()
160 {
161   lpos=rpos=2;
162   analogWrite(lme,lsp);
163   digitalWrite(lm2,HIGH);
164   digitalWrite(lm1,LOW);
165   analogWrite(rme,rsp);
166   digitalWrite(rm2,HIGH);
167   digitalWrite(rm1,LOW);
168 }
169
170 void orient(float st)
171 {
172   if((st-theta)>0.1)
173     {
174       if((st-theta)<=PI)
175       {
176         while((theta<(st-0.1))||(theta>(st+0.1)))
177           {
178             leftturn();
179             if (millis() - pre > 50)
180             {
181               computePosition();
182               pre = millis();
183             }
184             Serial.print(st);Serial.print(" ");
```

```
185             Serial.println(theta);
186           }
187           halt();
188       }
189       else if((st-theta)>PI)
190       {
191         while((theta<(st-0.1))||(theta>(st+0.1)))
192           {
193             rightturn();
194             if (millis() - pre > 50)
195             {
196               computePosition();
197               pre = millis();
198             }
199             Serial.print(st);Serial.print(" ");
200             Serial.println(theta);
201           }
202           halt();
203       }
204     }
205
206     else if((st-theta)<(-0.1))
207     {
208       if((st-theta)>=PI)
209       {
210         while((theta<(st-0.1))||(theta>(st+0.1)))
211           {
212             leftturn();
213             if (millis() - pre > 50)
214             {
215               computePosition();
216               pre = millis();
217             }
218             Serial.print(st);Serial.print(" ");
219             Serial.println(theta);
220           }
221           halt();
222       }
223       else if((st-theta)<PI)
224       {
225         while((theta<(st-0.1))||(theta>(st+0.1)))
226           {
227             rightturn();
228             if (millis() - pre > 50)
229             {
230               computePosition();
231               pre = millis();
232             }
233             Serial.print(st);Serial.print(" ");
234             Serial.println(theta);
235           }
236           halt();
237       }
238     }
239     reached=1;
240 }
241
242 float setang(float sx,float sy)
243 {
244   float ang;
245   if(((sy-yc)>=0)&&((sx-xc)>=0))
246   {
247     ang=atan((sy-yc)/(sx-xc));
248   }
249   else if(((sy-yc)>=0)&&((sx-xc)<=0))
250   {
```

```
    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);
```

```
317
318   wl = (leftTicks - leftTicksPrev) * c;
319   wr = (rightTicks - rightTicksPrev) * c;
320
321   leftTicksPrev = leftTicks;
322   rightTicksPrev = rightTicks;
323 }
324
325 void computePosition()
326 {
327   computeAngularVelocities();
328   unsigned long dt_integration = getElapsedTime(prevIntegrationTime);
329   prevIntegrationTime = micros();
330
331   float dt = dt_integration / 1000000.0;
332   float Vl = wl * RADIUS;
333   float Vr = wr * RADIUS;
334   float v = (Vr + Vl) / 2.0;
335   float w = (Vr - Vl) / LENGTH;
336
337   float thetaNext = theta + dt * w;
338   float xNext = xc + (dt * v*cos(thetaNext))/10.0;
339   float yNext = yc + (dt * v*sin(thetaNext))/10.0;
340
341   float distance = sqrt(xc * xc + yc * yc);
342
343   xc = xNext;
344   yc = yNext;
345   theta = thetaNext;
346
347   if(theta>2*PI)
348     theta=(theta-(2*PI));
349   else if(theta<0)
350     theta=(theta+(2*PI));
351
352   if(xc<0)
353   {
354     x1=(int(-1*xc))%100;
355     x2=(int(-1*xc/100))%100;
356     x3=(int(-1*xc/10000))%100;
357     xp=1;
358   }
359   else
360   {
361     x1=(int(xc))%100;
362     x2=(int(xc/100))%100;
363     x3=(int(xc/10000))%100;
364     xp=0;
365   }
366
367   if(yc<0)
368   {
369     y1=(int(-1*yc))%100;
370     y2=(int(-1*yc/100))%100;
371     y3=(int(-1*yc/10000))%100;
372     yp=1;
373   }
374   else
375   {
376     y1=(int(yc))%100;
377     y2=(int(yc/100))%100;
378     y3=(int(yc/10000))%100;
379     yp=0;
380   }
381
382   theta1=(int(theta*100))%100;
```

```
383   theta2=(int(theta*100))/100;
384 }
385
386 void lpid()
387 {
388   ldif=setlrpm-lrpm;
389   //lid+=ldif;
390   //ldd=ldif-lprev;
391   //lprev=ldif;
392   lsp=lpre_ocr+((255.0/400.0)*(ldif));
393   lpre_ocr=lsp;
394   if(lsp>255)
395     lsp=255;
396 }
397
398 void rpid()
399 {
400   rdif=setrrpm-rrpm;
401   //rid+=rdif;
402   //rdd=rdif-rprev;
403   //rprev=rdif;
404   rsp=rpre_ocr+((255.0/400.0)*(rdif));
405   rpre_ocr=rsp;
406   if(rsp>255)
407     rsp=255;
408 }
409
410 void payload_update()
411 {
412   payload[0] = xp;
413   payload[1] = x3;
414   payload[2] = x2;
415   payload[3] = x1;
416   payload[4] = yp;
417   payload[5] = y3;
418   payload[6] = y2;
419   payload[7] = y1;
420   payload[8] = theta2;
421   payload[9] = theta1;
422 }
423
424 void loop()
425 {
426   lrot=lctr/30.0;
427   rrot=rctr/30.0;
428
429   xbee.readPacket();
430
431   if (xbee.getResponse().isAvailable())
432   if(xbee.getResponse().getApiId()==ZB_RX_RESPONSE)
433   {
434     xbee.getResponse().getZBRxResponse(rx);
435     sxp=rx.getData(0);
436     sx3=rx.getData(1);
437     sx2=rx.getData(2);
438     sx1=rx.getData(3);
439     syp=rx.getData(4);
440     sy3=rx.getData(5);
441     sy2=rx.getData(6);
442     sy1=rx.getData(7);
443     stheta2=rx.getData(8);
444     stheta1=rx.getData(9);
445
446     if(sxp==0)
447     {
448       sx=sx3*10000+sx2*100+sx1;
```

```
449     }
450     else
451     {
452       sx=-1*(sx3*10000+sx2*100+sx1);
453     }
454     if(syp==0)
455     {
456       sy=sy3*10000+sy2*100+sy1;
457     }
458     else
459     {
460       sy=-1*(sy3*10000+sy2*100+sy1);
461     }
462     stheta=(stheta2*100.0+stheta1)/100.0;
463
464     Serial.print(sx);Serial.print(" ");
465     Serial.print(sy);Serial.print(" ");
466     Serial.println(stheta);
467
468     reach(sx,sy);
469     payload_update();
470     xbee.send(zbTx);
471   }
472
473   if (millis() - prevPositionComputeTime > 50)
474   {
475     computePosition();
476     prevPositionComputeTime = millis();
477   }
478
479   if( millis() - start > 200)
480   {
481     t=millis()-start;
482     start=millis();
483     lrpm=((lpulse/30.0)*60.0*1000.0)/t;
484     rrpm=((rpulse/30.0)*60.0*1000.0)/t;
485     lpulse=rpulse=0;
486   }
487 }
488
489 ISR(INT1_vect)
490 {
491   lctr++;
492   lpulse++;
493   if(lpos==2)
494     leftTicks--;
495   else
496     leftTicks++;
497   delay(10);
498 }
499
500 ISR(INT0_vect)
501 {
502   rctr++;
503   rpulse++;
504   if(rpos==2)
505     rightTicks--;
506   else
507     rightTicks++;
508   delay(10);
509 }
```

6.1.3 Laptop

```
1 #include <XBee.h>
2
```

```
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(' ')
```

```
41      xc1 = float( arr[0] )
42      yc1 = float( arr[1] )
43      thetac1 = float( arr[2] )
44      xc2 = float( arr[3] )
45      yc2 = float( arr[4] )
46      thetac2 = float( arr[5] )
47      x1.append(xc1)
48      y1.append(yc1)
49      theta1.append(thetac1)
50      x2.append(xc2)
51      y2.append(yc2)
52      theta2.append(thetac2)
53      drawnow(plot)
54      ctr=ctr+1
55      if(ctr>200):
56          x1.pop(0)
57          y1.pop(0)
58          theta1.pop(0)
59          x2.pop(0)
60          y2.pop(0)
61          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