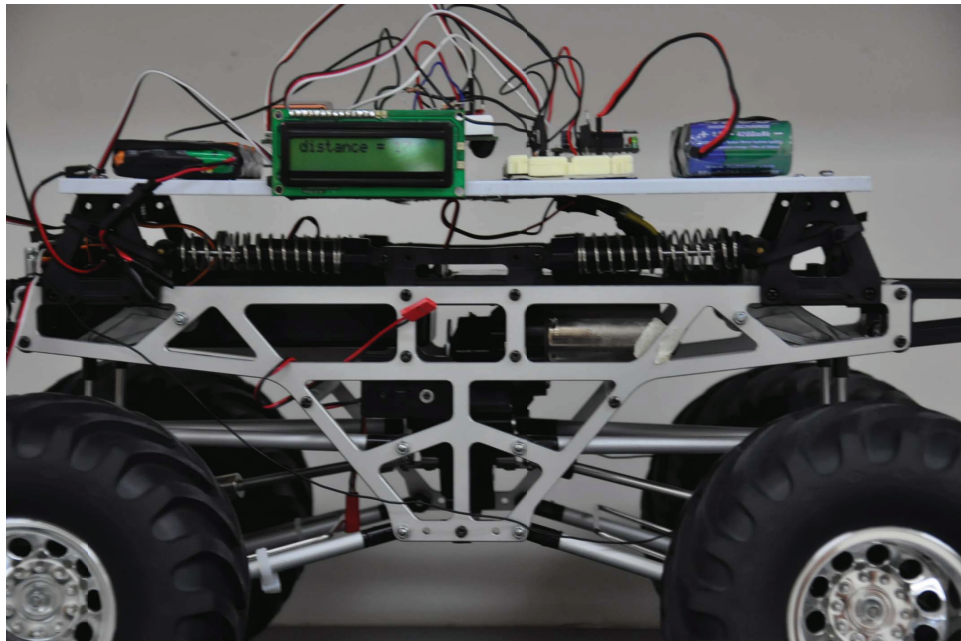


# Mechatronics Term Project

Karl Abdelnour, Robert Eckhardt, and Saumil Parakh

December 20, 2009



# Contents

<b>1</b>	<b>Project Outline</b>	<b>3</b>
<b>2</b>	<b>General Design</b>	<b>3</b>
<b>3</b>	<b>Hardware</b>	<b>3</b>
<b>4</b>	<b>Hardware Operation Principles</b>	<b>4</b>
4.1	Memsic 2125 . . . . .	4
4.2	Xbee-PRO . . . . .	5
4.3	Ultrasonic “Ping” Sensor . . . . .	5
4.4	LCD . . . . .	5
<b>5</b>	<b>Detailed Hardware</b>	<b>7</b>
<b>6</b>	<b>Numerical Scheme</b>	<b>7</b>
<b>7</b>	<b>PBasic Code-Appendix</b>	<b>9</b>
<b>8</b>	<b>Matlab Code-Appendix</b>	<b>9</b>

## Preface

The authors would like to thank the generous and gracious assistance of Dr. Vikram Kapila and Chandresh Dubey for their advice and guidance throughout the duration of this project and the course. The authors would further like to acknowledge thoughtful discussions with colleagues Parth Kumar, Kwok Yu Mak and, Ryan Caeti.

# 1 Project Outline

The 2010 Society of Automotive Engineers (SAE) Aero Design West competition brings together universities from various parts of the world to compete in a “heavy lift” competition. The goal being, to see which team can construct a plane, under given restrictions, that can lift the heaviest payload. The Mechanical Engineering Department at NYU-Poly has a history in competing at this event, and last year placed second for the first time in its history. One of the members of this mechatronics groups serves as a graduate advisor and team member in this year’s entry into the competition. New requirements from SAE mandate that aircraft have a data acquisition system (DAS) that is capable of measuring the takeoff distance of a competition aircraft, and display the distance on an LCD screen placed on the fuselage of the aircraft. This team has taken upon itself to design such a data acquisition system. The goal of this project is three fold:

1. To design a DAS capable of measuring the takeoff distance of the aircraft, and displaying it on an LCD screen on the plane. The initial design will be a “proof of concept” which will be tested on a remote control (RC) car.
2. To full-fill the course requirements of Mechatronics, and complete the term project.
3. To eventually adapt this “proof of concept” to the competition aircraft, once it is completed.

# 2 General Design

The project was designed under the premise that an accelerometer would be sufficient to complete the projects goals. The advantage of using an accelerometer versus other hardware solutions was its extendability and adaptability to other needs that may arise during the construction, design, and implementation of the design aircraft. Additionally, implementation of an accelerometer is minimally invasive to the overall architecture of the aircraft, and therefore can be the most easily modified and adjusted to account for changing system parameters. Consistency of accelerometer data was considered more important than the precision of the final result. Upon implementation more accurate filtering techniques can be implemented giving an accuracy that will approach the level of precision.

The basic stamp microcontroller (BS2), was a requirement for the mechatronics term project and, as a result, was the main component that bound all the subsidiary electronics together. PBasic, the basic stamp’s compiler language, is unable to robustly handle decimal point arithmetic, therefore, a wireless data solution was implemented. The XBee Pro RF module was chosen as the modem to handle the wireless communication. The RF device interfaced data logged into the basic stamp’s electronically erasable programable ROM (EEPROM) and wirelessly transmitted it to an Xbee base station located at a pc within its transmission range. The Xbee base station was integrated with Matlab R2007a such that the information received on the Xbee was then serially read into Matlab program. Matlab then filtered the data and made the appropriate numerical integrations using a discrete integration method with appropriate initial conditions, in order to convert the acceleration data into a position. This value was then sent wirelessly back to the Xbee located on the RC car, where it was displayed on an LCD screen on the side of the RC car.

The beginning of data transmission is user triggered; whereas, the end of data transmission between the Xbee on the RC car and the Xbee on the base station is determined by an ultrasonic “ping” sensor located on the front of the RC car. When the car lifts off the ground this simulates takeoff and prompts the system to stop numerically interpreting data. This process is described in more detail subsequently.

# 3 Hardware

1. BS2 Microcontroller
2. Memsic 2125 2 axis accelerometer
3. Xbee-Pro 802.15.4 OEM RF modules
4. Parallax Ultrasonic “Ping” sensor
5. Parallax 2x16 LCD screen

6. Tamiya 4x4 truck
7. Futaba S3005 high torque metal gear servomotor
8. Hobbico 1 way 8 cell 2A speed controller
9. 7.2V 4200mAh NiMH battery for the Xbee
10. 9.6V 2500mAh NiMH battery for the Tamiya RC car
11. JR 2.4GHz radio transmitter
12. Dell Precision M440 mobile workstation with Matlab R2007a

## 4 Hardware Operation Principles

### 4.1 Memsic 2125

The Memsic 2125 operates due to four temperature probes located within itself that monitor the relative location of a hot gas pocket within an enclosed chamber, see Figure 1. The Memsic 2125 converts these tem-

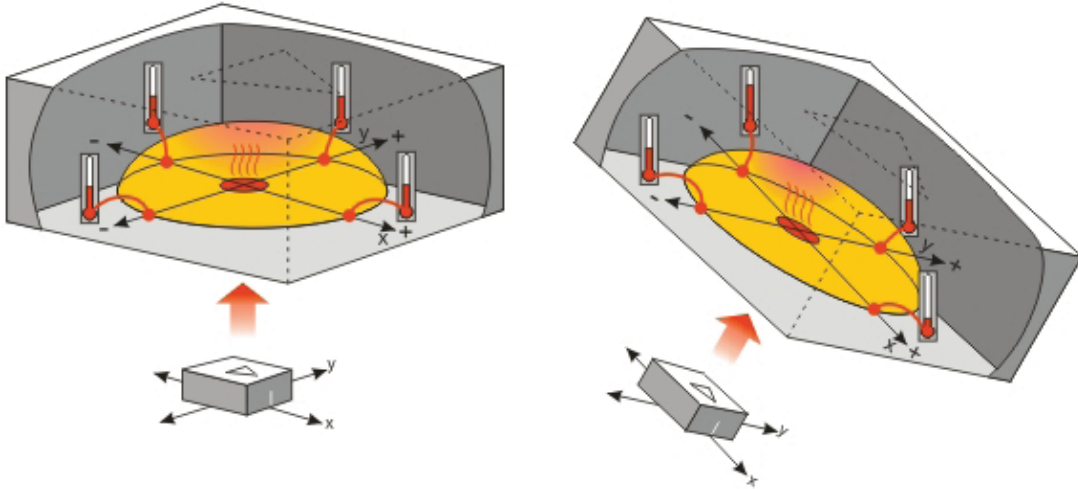


Figure 1: Memsic 2125 sensing principle

perature readings into pulse width modulation PWM signals which may then be converted to measurements of the force of gravity [1]. The scaling scheme to convert pulse widths are presented here.

$$\Gamma = K \cdot \left[ \frac{\alpha}{\beta - 1} \right] \quad (1)$$

where  $\Gamma$  is defined as the scale constant,  $\alpha$  is defined as the output scale elements, and  $\beta$  is defined as the input scale elements. Since we are concerned with displaying data in 1/100 g's, we choose  $-100$  to  $100$  as our desired output scale which corresponds to 201 scalable output elements. Since our accelerometer outputs data via its pulse width modulation in a range from 1875 to 3125, the difference of the range becomes the "input scale." This difference being 1251 if we include zero as an element. Disregarding zero as an element yields a difference of 1250.  $K$  is a constant value representing the amount of bits in a word, which is the variable storing the output data. The resulting  $\Gamma$  is recorded to the nearest integer.

Placing the following code within a loop, executes the `pulsin` command to retrieve data from the accelerometer and then scales it accordingly, a full replica of the implemented PBASIC code is provided in the appendix.

The pin out of the accelerometer is shown in the "Hardware Design" section of the report.

```

PULSIN 6, 1, y
PULSIN 7, 1, x

x = (x MIN 1875 MAX 3125) - 1875 ** 10538
y = (y MIN 1875 MAX 3125) - 1875 ** 10538

x_angle= (x MIN 1875 MAX 3125) - 1875 **13389 - 127
y_angle= (y MIN 1875 MAX 3125) - 1875 **13369 - 127

```

Figure 2: Snapshot of accelerometer scaling and acquisition PBasic code

## 4.2 Xbee-PRO

The Xbee also known as the “Zigbee” is a specification for a suite of high level communication protocol devices based on the IEEE 802.15.4-2003 standard for wireless personal area networks (WPANs). The Zigbee is intended to be a simpler, lower data rate, and higher battery life implementation of similar WPAN solutions such as bluetooth. Zigbee is a form of RF transmission.

The Xbee is programmed using the X-CTU software package as a “remote” and a “base” module. These modems are given identification tags so that they only communicate directly with each other. The Xbee is interfaced serially with Matlab R2007a, such that data may be wirelessly read into the base module and directly communicated to matrices and arrays within the numerical environment. The Matlab code to serially read in data is subsequently presented in Figure 6, and a complete printout of the code is provided in the appendix.

```

PULSIN 6, 1, y
PULSIN 7, 1, x

x = (x MIN 1875 MAX 3125) - 1875 ** 10538
y = (y MIN 1875 MAX 3125) - 1875 ** 10538

x_angle= (x MIN 1875 MAX 3125) - 1875 **13389 - 127
y_angle= (y MIN 1875 MAX 3125) - 1875 **13369 - 127

```

Figure 3: Matlab code used to serially read in the wirelessly transmitted data

## 4.3 Ultrasonic “Ping” Sensor

The Basic Stamp pulsin command uses a variable to store how long the high signal from the ping sensor lasts on the pulse width modulation. Knowing this high time and the speed of sound in air 344.8m/s the distance from the ping sensor to the object in its vicinity may be determined. The general operating equation governing the dynamics of the sensor are seen in Eq. (2)

$$D = \frac{1}{2}C \cdot T_{raw} \quad (2)$$

where,  $C \triangleq 344.8\text{m/s}$  and  $T_{raw}$  is the output time that the Basic Stamp delivers to the user in  $2\mu\text{s}$  units. The code used to implement the ping sensor is shown in Figure 4.3

## 4.4 LCD

The 2x16 LCD screen was used to visually display the position data on the side of the RC car. Serial communication between Matlab, the BS2, and the LCD screen were used in order to relay the information from one source to another. It is important to note that the baud rates need to be matched in order for this

```

PULSOUT 15, 5
PULSIN 15, 1, time
cmDistance = CmConstant ** time
DEBUG HOME, DEC3 cmDistance, " cm"

```

Figure 4: Snapshot of code used with ultrasonic sensor

```

%decimal to string and write to pbasic

str_x = num2str(position);
fprintf(porta_ser,str_x)

%close serial port
fclose(porta_ser);
delete(porta_ser);

```

Figure 5: Snapshot of code used to print data from Matlab to the BS2

to happen. The formula for computing baud rate is shown to be:

$$\zeta = \frac{1 \cdot 10^6}{B} - 20 \quad (3)$$

where  $\zeta$  in Eq. (3) refers to the constant that PBasic interprets to determine which baud rate its using. For the purposes of our serial communication a Baud rate of 38,400 was selected. This corresponded to a PBasic constant of 6. It is simpler to transmit the data as a string, rather than a decimal and as a result the “num2string” function was used in Matlab to convert the data accordingly. Figure 16, shows how the data is written from Matlab to the serial port and therefore received by the Xbee on the remote base.

The path of data from Matlab to the LCD goes from the PC to serial port COM6, to the Xbee located on the RC car, to the Basic Stamp and finally to the LCD Screen.

In Figure ?? RX refers to the pin assignment in which PBasic reads the input, Baud is the baud rate of the communication between Xbee’s (38400) and sData is the variable the data is stored in. The subsequent code in the same figure shows how data is communicated from Pbasic to the LCD screen. 84 is the constant  $\zeta$  computed as shown in Eq. (3). The LCD screen supports a baud rate of 9600 and this corresponded to a  $\zeta$  of 84.

```

'Reads data from Matlab writes Data to LCD

SERIN RX,Baud, [SDEC sData]

'write to lcd whats received
SEROUT 14,84, [22,12] 'serout to pin 14 at 9600 baud, 22 turns LCD on, and 12 clears the display
PAUSE 5
SEROUT 14,84, ["distance = ",DEC sData]      '13 is a carriage return in ASCII

DEBUG ? sData

DEBUG CR, "Press Enter to exit..."
DEBUGIN char

```

Figure 6: Snapshot of code used to read in data from remote Xbee and print to LCD

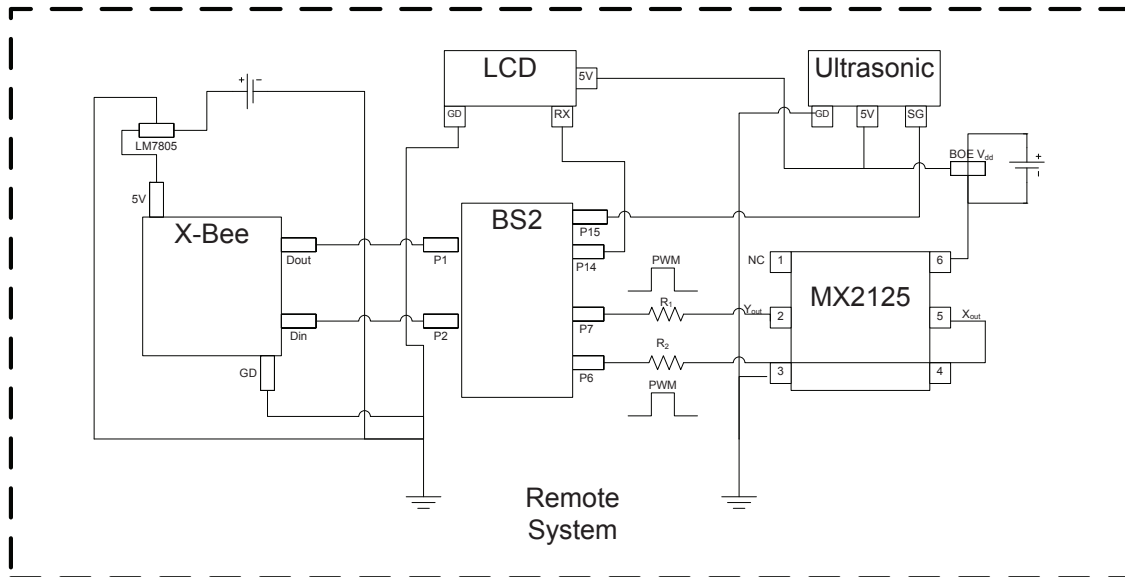


Figure 7: Remote Circuit Diagram on RC Car

## 5 Detailed Hardware

The circuit diagram of the components used in our mechatronics project are presented here, in Figure 7. The corresponding circuit diagram of the components on the PC side base are shown here, in Figure 13.

## 6 Numerical Scheme

In order to process the data, it needed to be filtered and then integrated in Matlab. Once the data had been read in serially, it needed to be smoothed. A moving average smoothing algorithm using the “smooth” function in Matlab was implemented after which the filtered data was numerically integrated using a Backwards Euler discrete integration scheme.

Position may be determined by twice integrating the acquired acceleration data given zero velocity and position initial conditions.

$$v = \int a dt \quad (4a)$$

$$x = \int v dt \quad (4b)$$

where,  $v$  and  $x$  denote the analog values of position and velocity obtained via integration over a continuous time space. Since we are dealing with digital sensors and as a result discrete data, numerical integration techniques were employed.

The frequency of acquisition of the accelerometer is known and as a result, the time step  $\Delta t$  between acquired data points is known. By performing the following discrete integrations.

$$v_i = v_{(i-1)} + a_i \Delta t \quad (5a)$$

$$s_i = s(i-1) + v_i \Delta t \quad (5b)$$

where  $v_0 = 0$  and  $s_0 = 0$ . The full code of the smoothing algorithm and numerical integration scheme is provided in the appendix.

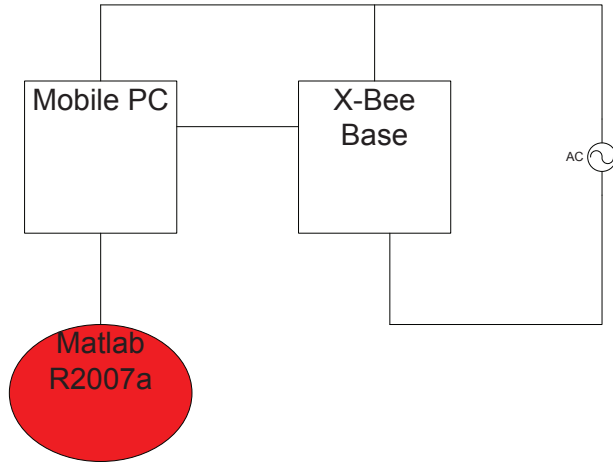


Figure 8: Remote Circuit Diagram on RC Car

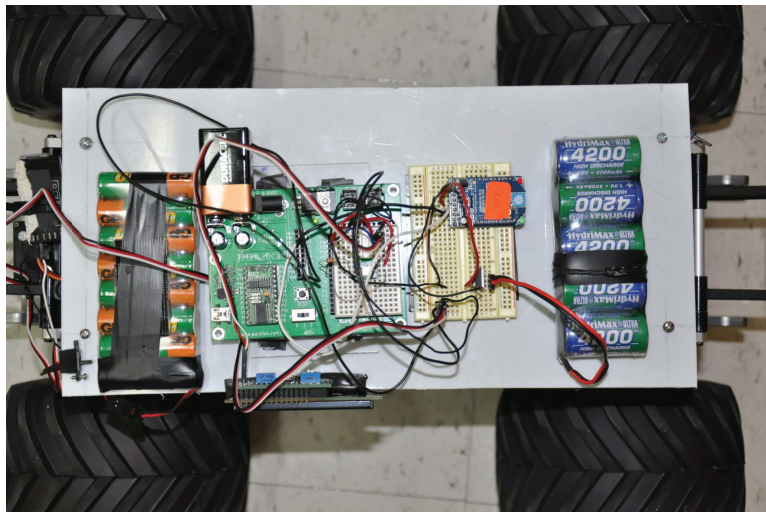


Figure 9: Circuit on the RC Car



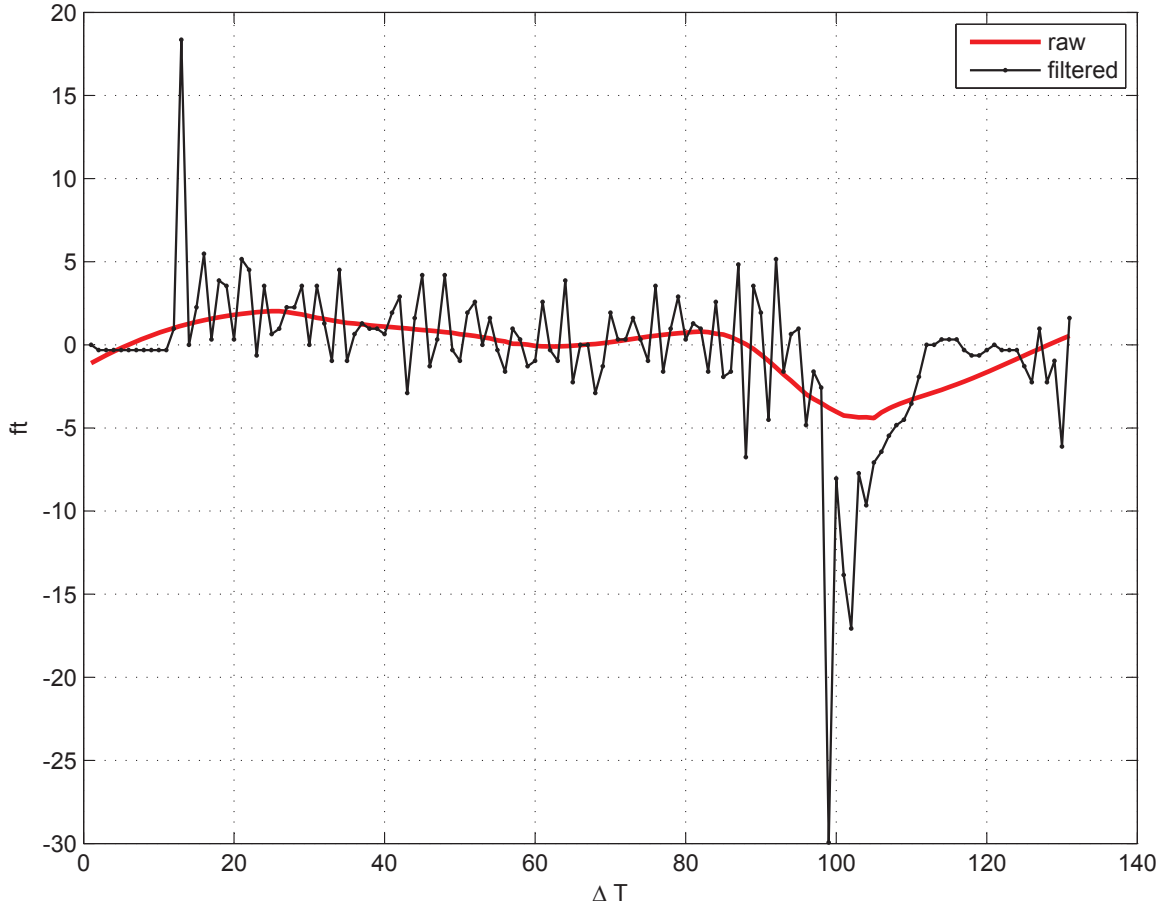


Figure 10: Raw then filtered data

## 7 PBasic Code-Appendix

The preamble of the code begins by declaring all appropriate variables. DATA is a functional variable which stores records directly to the eeprom of the BS2. The eeprom is extra memory available to the basic stamp which is ideal for data logging applications.

The next section of the code deals with assembling the acceleration data from the pulse widths outputted from the MX2125. The pulsln command is used to obtain data from the BS2. The scaling parameters are applied as previously discussed. Additionally, the ultrasonic sensor data is acquired at the same frequency and analyzed in Matlab to determine when the RC car lifts off the ground.

The last portion of the PBasic code outputs the data logged data from the eeprom to the serial port in which the Xbee is connected. Data is read from the eeprom and then using the serout command and appropriate baud rate is communicated to the serial port COM6. A terminator signal is then sent to the serial port to allow Matlab to close its serial port and start processing the received data.

The program then has a serin command which waits for data to be sent back from Matlab to the Basic Stamp with the processed results.

## 8 Matlab Code-Appendix

The Matlab code consists of several, components. The first creates a serial object porta. This opens COM6 on the PC for data entry. The timeout size is made sufficiently large such that data can have enough time to fully stream through. A terminator, the ASCII character 'A' is made such that when it receives this character from PBasic it terminates the data acquisition portion of its program and moves on to the smoothing algorithm.

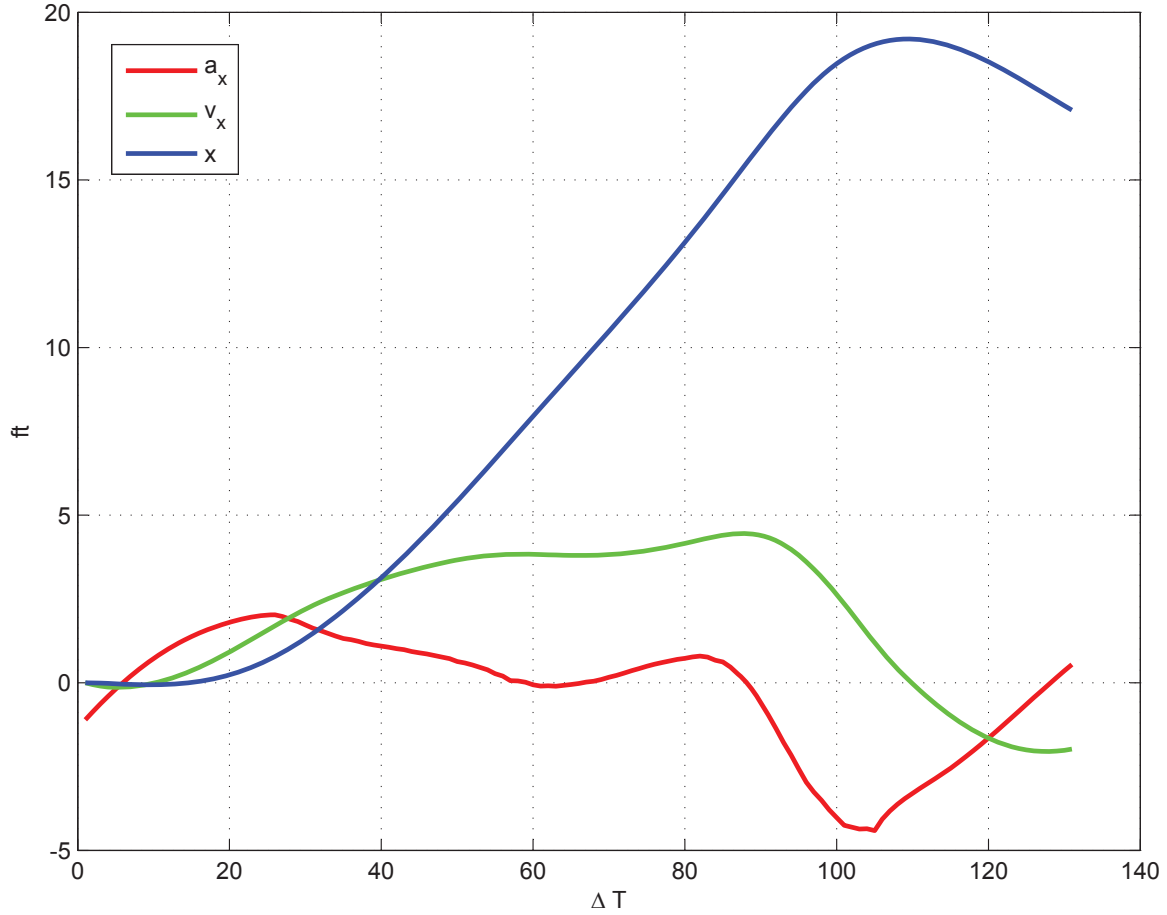


Figure 11: Plot of acceleration, velocity and position

The Matlab smooth function is used to find a moving average fit line of the experimentally obtained data points. The smoothed acceleration data is fed into the numerical integration scheme outlined in Eq. (5). The final position data is then converted to a string and then set to PBasic using the fprintf command. The code is presented subsequently.

## References

- [1] Dr. Vikram Kapila. Polytechnic Institute of New York University, 6 MetroTech Center, Brooklyn NY, 11201, 2009. Mechatronics Class Notes.

```

' {$STAMP BS2}
' {$PBASIC 2.5}

'Declare variables and constants for MX2125

x      VAR Word
y      VAR Word
side  VAR Word
sign  VAR Bit
counter VAR Byte
xlin  VAR Nib
Negative CON 1
sData VAR Word
CmConstant CON 2260
cmDistance VAR Word
time  VAR Word
eeIndex    VAR    Word
Records    DATA  (500)
RecordsEnd  DATA
char       VAR    Byte
'-----
'Declare variables and constants for serial communication with Xbee

Baud      CON 6      ' Baud rate, 38400, 8-N-1, non-inverted, on BS2

RX        PIN 1      ' DOUT
TX        PIN 2      ' DIN

```

Figure 12: Remote Circuit Diagram on RC Car

```

'-----
'Record acceleration and angle of heading

FREQOUT 4, 75, 4000
PAUSE 200
FREQOUT 4, 75, 4000

DEBUG CLS, "Acquiring x_acceleration, y_acceleration and angle"

FOR eeIndex = Records TO RecordsEnd STEP 2

    PULSIN 6, 1, y
    PULSIN 7, 1, x

    x = (x MIN 1875 MAX 3125) - 1875 ** 10538
    y = (y MIN 1875 MAX 3125) - 1875 ** 10538

PULSOUT 15, 5
PULSIN 15, 1, time
cmDistance = CmConstant ** time
DEBUG HOME, DEC3 cmDistance, " cm"

WRITE eeIndex, x
WRITE eeIndex + 1, cmDistance
'WRITE eeIndex + 2, y

NEXT

FREQOUT 4, 200, 4000

```

Figure 13: PBasic Code-acceleration

```

DEBUG CR, " Index  x-axis  y-axis  angle",
      CR, "-----  -----  -----  -----",
      CR
FOR eeIndex = Records TO RecordsEnd STEP 2
  READ eeIndex, x
  x = x - 100
  READ eeIndex + 1, cmDistance
  cmDistance = cmDistance
  'READ eeIndex +2, y
  'y= y - 100
  ' DEBUG DEC eeIndex, CRSRX, 7, SDEC x, CRSRX, 14, SDEC y,CRSRX, 21, SDEC angle,CRSRX, 28,SDEC
angleh, CR

  SEROUT TX,Baud, [SDEC x, CR]
  SEROUT TX,Baud, [SDEC cmDistance, CR]

  'SEROUT TX,Baud, [SDEC y,CR]
  'PAUSE 50 'serout delay

NEXT

'terminator
SEROUT TX, Baud, [65]

'-----
'Reads data from Matlab writes Data to LCD

SERIN RX,Baud, [SDEC sData]

'write to lcd whats received
SEROUT 14,84, [22,12] 'serout to pin 14 at 9600 baud, 22 turns LCD on, and 12 clears the display
PAUSE 5
SEROUT 14,84, ["distance = ",DEC sData]      '13 is a carriage return in ASCII

DEBUG ? sData

DEBUG CR, "Press Enter to exit..."
DEBUGIN char

```

Figure 14: PBasic Code-wireless

```

clear all; close all; clc;

% %read in data from Xbee
porta_ser = serial('COM6');
porta_ser.BaudRate = 38400;
porta_ser.DataBits = 8;
porta_ser.Parity = 'none';
porta_ser.StopBits = 1;
porta_ser.FlowControl = 'none';
porta_ser.InputBufferSize = 10000000;
porta_ser.Timeout=1000;
porta_ser.Terminator='A';

fopen(porta_ser);
pause(.001);
out=fscanf(porta_ser,'%i');

%-----
%Sort acceleration data
a=out(1:2:end);
b=out(2:2:end);

accel=[a(1:length(b)),b]

r=length(b) %#ok<NOPTS>
for ii=1:r
    if (b(ii)> 20)
        r=ii %#ok<NOPTS>
        break
    end
end

a_new=[a(1:r),b(1:r)];

%-----
%% Butterworth filter low pass

Fs = 15; % acquisition frequency in Hz
% HFs = Fs/2;
% fNorm = 1/HFs;
a_x = a_new(:,1)*32.2/100;
%
xi= length(a_x);
% [b,a] = butter(10, .5, 'low'); %the order can be changed
% %A = firlt(1/Fs*[1:xi],b, a, a_x);
A=smooth(1/Fs*[0:xi-1],a_x,0.1,'rloess');

%Integration routine
%length of time step
DT=1/(Fs);
V0=0.;

```

Figure 15: Matlab code

```
V(1)=V0; %Forward Euler
for ii=2:xi
    V(ii) = V(ii-1) + A(ii)*DT; %Velocity
end

s0=0.; %forward Euler
s(1)=s0;
for ii=2:xi
    s(ii) = s(ii-1) + V(ii)*DT; %position
end
s(xi);
distance=s(xi)
position=floor(distance)

plot(A, '-r')
hold on
% plot(a_x, '--k')
plot(V, 'g-')
plot(s, '-b')
hold off

%-----
%decimal to string and write to pbasic

str_x = num2str(position);
fprintf(porta_ser,str_x)

%close serial port
fclose(porta_ser);
delete(porta_ser);
```

Figure 16: Matlab code