Abbreviated Lab Report

Lab #1: Serial Communication

Prepared by: \_\_\_\_Sean Lantto\_\_\_\_

Team Members: \_\_\_\_\_\_Giovani Rodriguez Leon, Jordan Minshall\_\_\_\_\_\_\_\_\_\_\_

02/08/2017

1. **Objective**

* Learn about the basic serial communication protocol;
* Learn how to send and receive serial communication packets from MATLAB;
* Practice MATLAB programing.

1. **Observation and Discussion:**

Using a null modem cable enabled the communication between two ports on the computer. It simply crosses the tx from one COM port over to the rx of the other and the rx of the first COM port to the tx of the other. This prevents the need for having 2 opposite serial connecters on a computer to enable self communication or communication with another computer. In the case of his lab, a usb to dual DB-9 cable was used and then a null modem connector was used to join the two DB-9 connectors. This acted like a connection between two COM ports while only using a single USB port.

With the connecton between the two serial ports physically connected, two instances of the program MTTTY were opened, and the each assigned a different COM port (2 and 19 in this case), and were set to use the same Baud rate of 115200 bps. The communication between the two ports was tested by sending different strings of letters. Once the connection was confirmed one of the MTTTY instances was terminated, and an instance of MATLAB was opened. The source code for the MATLAB is in Appendix B below. In MATLAB a 10 bit packet (shown in the table in Appendix A) was assembled and sent out in a continuous stream of 1024 packets, that was received by the MTTTY program. It was confirmed that packets were received in MTTTY, which displayed the bytes as ASCII characters as opposed to their actual numerical value.

MTTTY was closed, and a code to receive and decode the previously created data packets was written. The results from the receiving MATLAB code are shown in the figures in Appendix A. It is interesting to note the strange deformation in near the end of the time plot, and counter plot for the code running with a 0.01 second pause. Its cause is unknown.

**Answers to deliverable questions:**

1. Answered above in the first paragraph

2. To display 21.49 on MTTTY you would need to input binary values [50 49 46 52 57]

3. The serial buffer holds data while being transmitted or received, and usually operates as a First-In, First-Out (FIFO)system.

4. During the Lab, it seemed that no packets were lost despite changes in the pause value. However before changing the buffer size, there was a 100% loss rate, or more specifically corruption rate, as they were partial packets. Increasing the buffer minimized packet loss to 0.

5. It is possible to quantify the communication delay, one way could be to compare clocks between the time in the packet to the time it was received. To minimize the delay, the serial buffer should not be made too large.

1. **Problems and Solutions**

The largest issue encountered was the corruption of all the packets when transmitting between MATLAB Tx and Rx. This was discovered to be a result of the serial read function looking to read 8 bytes, when it should have been looking for 10.

Another issue encountered was the check sum being a constant 255 for all packets, but was solved by adjusting the code to properly sum all the packets.

1. **Learning**

I learned of the MATLAB serial library, which I had never seen before. While I have used serial communication in other applications (Arduinos, a Raspberry Pi, etc), I had never put any thought into the packet structure or buffers, and had always taken that flow of data for granted.

1. **Comments and Suggestions**

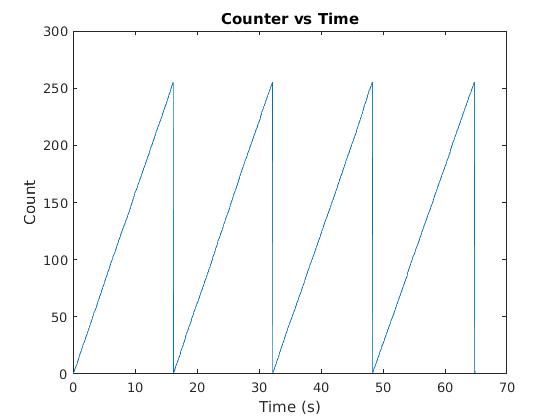
I think the lab could be improved upon by attempting communication between two computers, with one group writing the both a transmitting and receiving code. It would also be cool, to try and code data from a sensor into a packet, and send it from an arduino to the computer. Otherwise this was a good lab.

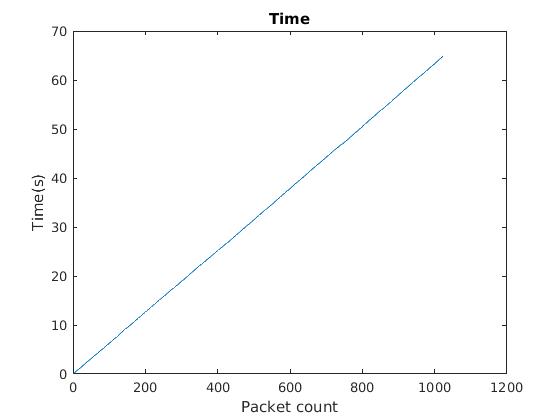
**Appendix A – Figures and Tables**

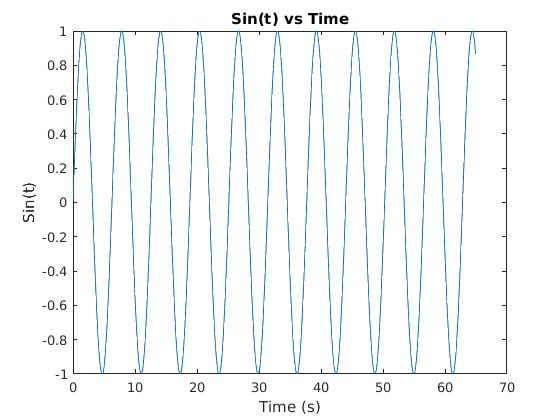
**Table showing structure of the data packet created**

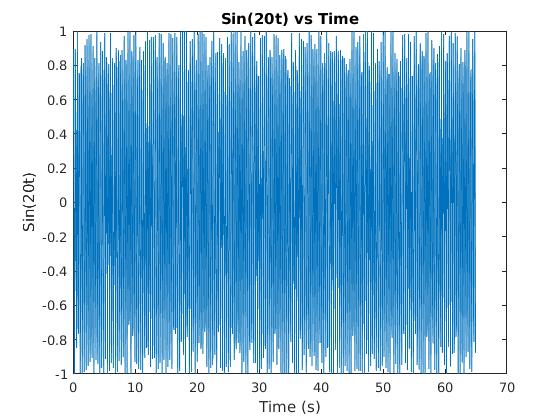
|  |  |  |
| --- | --- | --- |
| **Byte** | **Content** | **Comment** |
| 0 | Header 1 (65) | ASCII ‘A’ |
| 1 | Header 2 (90) | ASCII ‘Z’ |
| 2 | Counter | An 8-bit counter ranging from 0 to 255 |
| 3,4 | Time, t |  |
| 5,6 | Data Ch.1: sin(t) |  |
| 7,8 | Data Ch. 2: 20sin(t) |  |
| 9 | Check Sum |  |

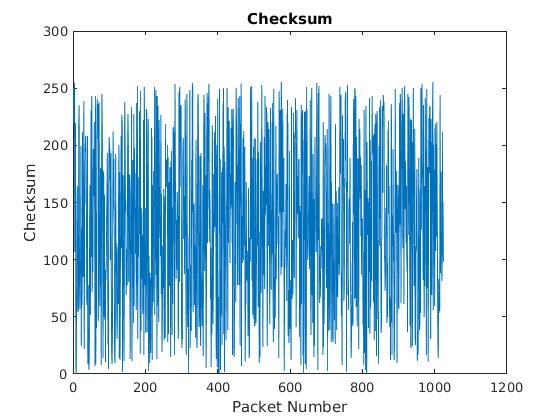
**Figures from MATLAB to MATLAB serial communication with 0.02 second pause:**

****

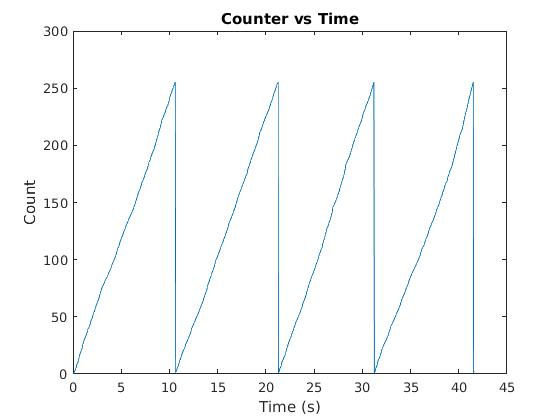
****

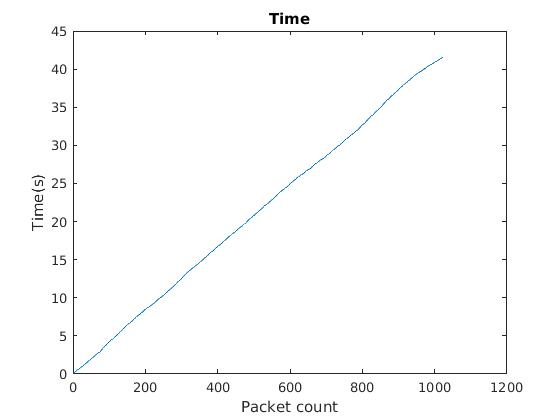
****

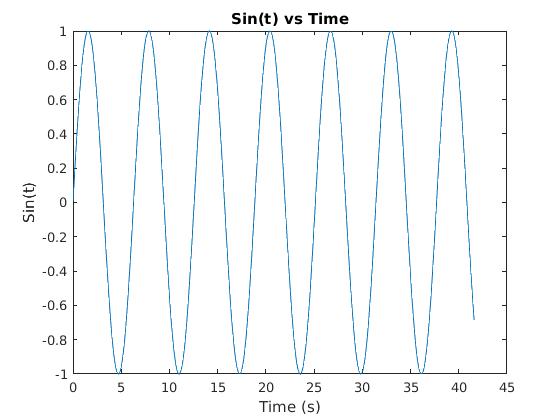
****

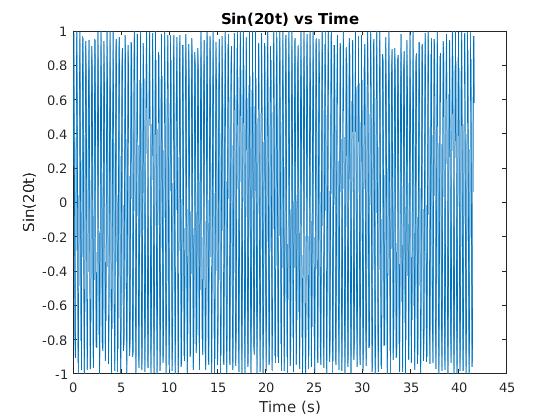
****

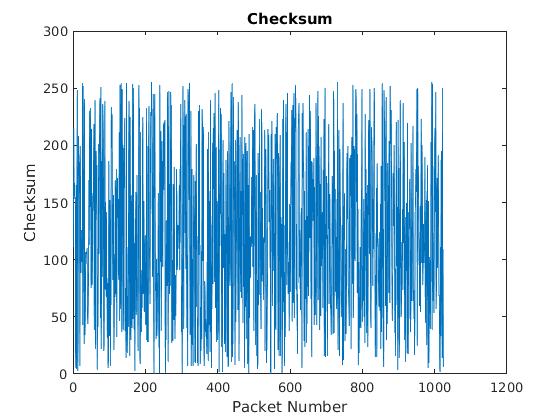
**Figures from MATLAB to MATLAB serial communication with 0.01 second pause:**

****

****

****

****

****

**Appendix B - Source Code**

Initialization code and testing the communication:

Cereal19=serial('COM19')

set(Cereal19,'BaudRate',115200)

fopen(Cereal19)

Cereal2=serial('COM2')

set(Cereal2,'BaudRate',115200)

set(Cereal2,'InputBufferSize',10)

fopen(Cereal2)

b='Mobile Robotics';

c=uint8(b)

fprintf(Cereal19,'Mobile Robotics ')

fscanf(Cereal19, '%s')

Data stream code

%Transmitter

pt=0.01;

stream = [];

Stream=[];

tic; %Time refrence

count = uint8(0); %Initialize byte counter

for n = 1:1024

pause(pt); %time pause

Head0 = uint8(65); %Headers

Head1 = uint8(90);

Byte0 = Head0;

Byte1 = Head1;

Byte2 = count; %ByteCounter

if count == 255 %Byte Counter reset

count = uint8(0);

else

end

t = toc; %take time

WholeT = t\*500; %Amplify time by a range usable

Byte34 = typecast(uint16(WholeT),'uint8'); %Convert time to uint16 then split into uint8

%Both Sin functions are aplified by 10000 converted to int16 then split

%into uint8

SINt = sin(t);

Whole56 = SINt \* 10000;

Byte56W = int16(Whole56);

Byte56= typecast(Byte56W,'uint8');

SIN2t = sin(20\*t);

Whole78 = SIN2t \* 10000;

Byte78W = int16(Whole78);

Byte78= typecast(Byte78W,'uint8');

%Sm is the sum for the checksum

Sm=double(Byte0)+double(Byte1)+double(Byte2)+double(Byte34(1))+double(Byte34(2))+double(Byte56(1))+double(Byte56(2))+double(Byte78(1))+double(Byte78(2));

Sm=uint8(mod(uint32(Sm),256));

Byte9 = Sm;

%for the packets and the stream

PKT= [Byte0 Byte1 Byte2 Byte34(1) Byte34(2) Byte56(1) Byte56(2) Byte78(1) Byte78(2) Byte9];

%stream = [stream PKT];

count = count +1;

fwrite(Cereal19,[PKT]);

PKTrecieved = fread(Cereal2);

Stream = [Stream transpose(PKTrecieved)];

end

%fwrite(Cereal19,[stream]);

%Jordan Minshall

%Reciever

Packet = [];

DATA = zeros(4,1024); %Preallocated data matrix

for p = 1:1024 %This counts packets

PackStart = 10\*(p-1)+1; %Determines the begging and ending indice of each packet

PackEnd = 10\*p;

if PackStart == 0

PackStart = 1;

else

end

j = 1;

for P = PackStart:PackEnd

Packet(j) = Stream(P); %forms the active packet

j = j +1;

end

CheckSumCount = 0;

for b = 1:10 %Cycles through the bytes of the active packet

Byte = Packet(b);

DecByte = double(Byte);

switch b

case 1 %Check first header

if DecByte == 65

fprintf('Packet %1.0f Header 1 correct, 65 \n',p)

else

fprintf('ERROR Packet %1.0f Header 1, %3.0f \n',p,DecByte)

end

case 2 %Check second header

if DecByte == 90

fprintf('Packet %1.0f Header 2 correct, 90 \n',p)

else

fprintf('ERROR Packet %1.0f Header 2, %3.0f \n',p,DecByte)

end

case 3 %Saves Counter for active packet

Counter255 = DecByte;

case 4 %Saves first time byte of active packet

TIME1 = DecByte;

case 5 %Saves second time byte of active packet

TIME2 = DecByte;

case 6 %saves 1st sin(t) byte of active packet

SINt1 = DecByte;

case 7 %saves 2nd sin(t) byte of active packet

SINt2 = DecByte;

case 8 %saves 1st sin(20t) byte of active packet

SIN20t1 = DecByte;

case 9 %Saves 2nd sin(20t) byte of active packer

SIN20t2 = DecByte;

case 10 %Checks checksum

if Byte == uint8(mod(uint32(CheckSumCount),256));

fprintf('Packet %1.0f CheckSum Correct \n',p)

else

fprintf('ERROR Packet %1.0f,CheckSum = %3.0f Count = %3.0f \n',p,DecByte,CheckSumCount)

end

ChkSum = uint8(Byte);

end

%adds bytes to verify checksum

CheckSumCount = CheckSumCount + DecByte;

end

%Combines data into in16

Time = typecast(uint8([TIME1 TIME2]),'uint16');

Sint = typecast(uint8([SINt1 SINt2]),'int16');

Sin20t = typecast(uint8([SIN20t1 SIN20t2]),'int16');

% converts to double then divides this was the error I had been

% expierencing when we saw each other in the lab I had been dividing an

% int16 and matlab was dropping the decimals

Time = double(Time)/500;

Sint = double(Sint)/10000;

Sin20t = double(Sin20t)/10000;

%archive packet data so the loop can continue to the next

DATA(1,p) = Counter255;

DATA(2,p) = Time;

DATA(3,p) = Sint;

DATA(4,p) = Sin20t;

DATA(5,p) = ChkSum;

DATA(6,p) = p;

end

plot(DATA(2,:),DATA(1,:));

title('Counter vs Time');

xlabel('Time (s)');

ylabel('Count');

figure

plot(DATA(2,:),DATA(3,:));

title('Sin(t) vs Time');

xlabel('Time (s)');

ylabel('Sin(t)');

figure

plot(DATA(2,:),DATA(4,:));

title('Sin(20t) vs Time');

xlabel('Time (s)');

ylabel('Sin(20t)');

figure

plot(DATA(6,:),DATA(2,:));

title('Time');

xlabel('Packet count');

ylabel('Time(s)');

figure

plot(DATA(6,:),DATA(5,:));

title('Checksum')

xlabel('Packet Number')

ylabel('Checksum')

save('Lab1TxRxPause01')