RFID Tracing

Yujing Gong, 4471458, Qiaoqiao Li, 4508300

March 24, 2016

1 Introduction

In this assignment, we plot and decode an RFID signal between Reader(R)/Interrogator and Tag (T) into binary format in MATLAB. The binary code is further explained. And the script used to decode the signal is provided in Appendix.

2 Algorithm

Signals between Reader and Tag have different modulation modes. According to Radio-Frequency Identity Protocols, signals sent by Reader could be decoded according to the width of each pulse while signals sent by Tag could be decoded according to the signal status during each time interval.

Firstly, MATLAB is used to import the input signal data and save them in an array variable called "data". Then the entire signal is plotted to give us a general view of the signal. The RFID signal consists of four clusters, the first and third are signals sent from Reader to Tag and the second and fourth are signals sent from Tag to Reader.

A simple algorithm is used in this realization of decoding. The basic idea of how we distinguish between "1" and "0" is briefly explained below.

As for the signal from R to T, only estimation of pulse width should be made to decode the signal. We use Figure 2 to show how the width of each pulse is calculated. For example, we want to calculate the width of the first pulse. This can be estimated by calculating the time between point B and C. B is recognized as the first data after point A that has a value larger than 0.4 and C is recognized as the first data after point B that has a value smaller than 0.3. The width of the first pulse can be calculated by subtraction of index of point B and point C. The reason why we use different values (0.3 and 0.4) to get point B and C is

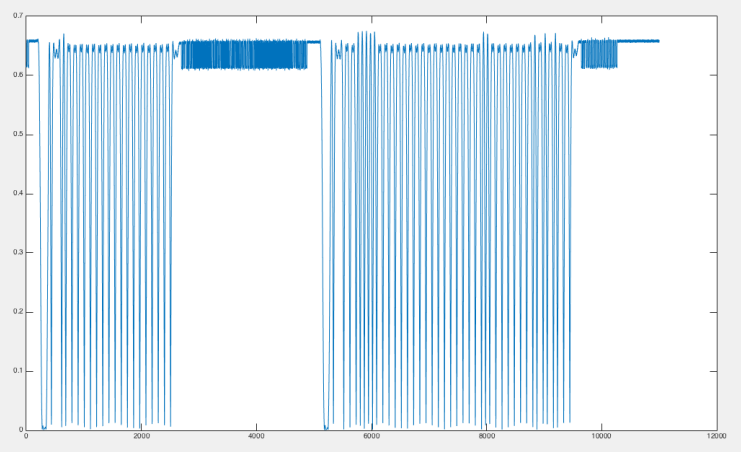

Figure 1: RFID signal.

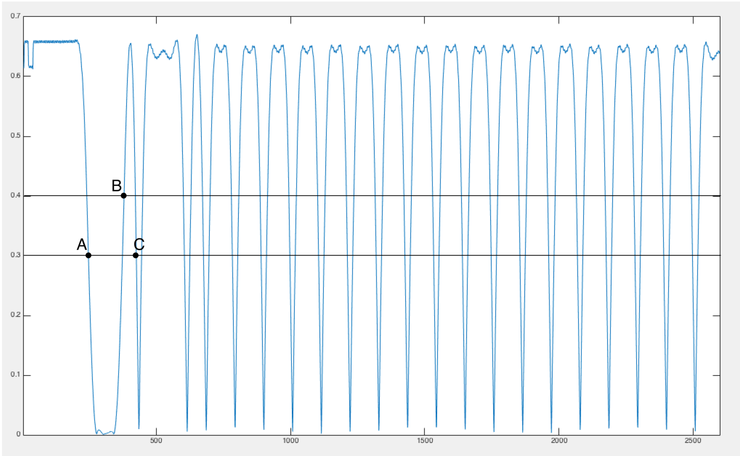

Figure 2: Pulse identification.

that we don't know if a signal is smooth or fluctuating. For example, if point B is found as the first data exceeding 0.3 after point A, this point B may not be the expected point if there is fluctuation around point A. It could be a fluctuating point around point A that has a value larger than 0.3. Hence, we use different values as thresholds to calculate the width of each pulse. We assume we get more accurate results in this way.

As for the signal from T to R, we need to see whether there is a change of signal level in each time interval. Same as processing signal from R to T, we can get an array of data index called "index2". For example, if the value of index2(i+1)-index2(i) is larger than a certain value (here we call this value "comp_width"), we consider the pulse between index2(i+1) and index2(i) as signal "0"; if the value of index2(i+1)-index2(i) is smaller than "comp_width", we consider this signal as "1". We then iteratively to compare index2(i+3)-index2(i+1) with "comp_width".

3 Decoding result

3.1 Signal to code

The decoding result is a stream of bits. They are divided into four pieces as follows:

First R ->T: 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

First T ->R: 0 0 1 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 1 0 0 1 0 1 1 0 0 0 0 0 0 0 0 0 1 1 1 0 0 0 1 0 0 0 0 1 0 1 0 1 1 0 1 1 1 1 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 0 0 0 1 0 1 1 0 0 1 0 1

Second R ->T: 1 1 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 1 1 1 1 1 1 1 0 1 0 1 0 1 1

Second T ->R: 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1

3.2 Explanation

1. The first 2 bits of "First R->T" are "01", indicating that it is an Interrogator Command of *ACK*. The length of this signal is 18 bits and reply is immediately needed. The following 16 bits are all ones and represent the *Echoed RN16 or handle*.

2. "First T->R" is the tag reply to the previous *ACK* command. The first 3 bits of it are "001", indicating that this tag reply is disallowed. Thus, the following bits make no sense.

3. The first 8 bits of "Second R->T" are "11000001", indicating that it is an Interrogator Command of *Req _ RN*. The length of this signal is 40 (bits) and reply is

immediately needed. Protection of this signal is *CRC-16*. The following 16 bits are all ones, which again represent *Prior RN16 or handle*. The next following 16 bits are "0011111110101011" and represent *CRC-16*.

4. The "Second T->" is the second tag reply. The first 16 bits of it are all ones and represent *handle or new RN16*. The last 16 bits are also all ones and represent *CRC-16*.

Appendix

```
data=[];
data=importdata('signal.txt');
plot(data);
x=1;k=1;i=1;begin=1;
signal1=2500;signal2=5000;signal3=9450;signal4=10350;
reader1=[];index1=[];reader2=[];index3=[];
tag1=[];tag2=[];index2=[];index4=[];

% get the 1ST signal from READER
for a=1:signal1;
    if data(a)<0.3;
        i=a;
        break;
    end
end

while i<signal1,
    if data(i)>0.4;
        index1(x)=i;
        x=x+1;
        for j=i:signal1;
            if data(j)<0.3;
                index1(x)=j;
                i=j;
                x=x+1;
                break;
            end
            if j==signal1;
                break;
            end
        end
    else
        i=i+1;
    end
end

for z=1:size(index1,2)-1;
    if index1(z+1)-index1(z) > 100;
        begin=z+2;
        break;
    end
end

for z=begin:2:size(index1,2)-1;
```

```
    if index1(z+1)-index1(z) > 60;
        reader1(k)=1;
    else
        reader1(k)=0;
    end
    k=k+1;
end

% get the 1ST signal from TAG
x=1;k=1;begin=1;t1_i=begin;
for a=signal1+100:signal2;
    if data(a)<0.62;
        i=a;
        break;
    end
end

while i<signal2,
    if data(i)<0.63;
        index2(x)=i;
        x=x+1;
        for j=i:signal2;
            if data(j)>0.64;
                index2(x)=j;
                i=j;
                x=x+1;
                break;
            end
            if j==signal2;
                break;
            end
        end
    else
        i=i+1;
    end
end

for z=1:size(index2,2)-1;
    if index2(z+1)-index2(z) > 22;
        begin=z+2;
        break;
    end
end

t1_i=begin;
while t1_i < size(index2,2),
    if index2(t1_i+1)-index2(t1_i) < 10;
        tag1(k)=0;
        t1_i=t1_i+2;
    elseif index2(t1_i+1)-index2(t1_i) >= 13;
        tag1(k)=1;
        t1_i=t1_i+1;
    else
        tag1(k)=2;
    end
```

```
    k=k+1;
end

% get the 2ND signal from READER
x=1;k=1;begin=1;i=1;

for a=signal2:signal3;
    if data(a)<0.3;
        i=a;
        break;
    end
end

while i<signal3,
    if data(i)>0.4;
        index3(x)=i;
        x=x+1;
        for j=i:signal3;
            if data(j)<0.3;
                index3(x)=j;
                i=j;
                x=x+1;
                break;
            end
            if j==signal3;
                break;
            end
        end
    else
        i=i+1;
    end
end

for z=1:size(index3,2)-1;
    if index3(z+1)-index3(z) > 100;
        begin=z+2;
        break;
    end
end

for z=begin:2:size(index3,2)-1;
    if index3(z+1)-index3(z) > 60;
        reader2(k)=1;
    else
        reader2(k)=0;
    end
    k=k+1;
end
figure;
plot(reader2);

% get the 2ND signal from TAG
x=1;k=1;begin=1;t2_i=begin;
for a=signal3+50:signal4;
    if data(a)<0.62;
```

```
        i=a;
        break;
    end
end

while i<signal4,
    if data(i)<0.63;
        index4(x)=i;
        x=x+1;
        for j=i:signal4;
            if data(j)>0.64;
                index4(x)=j;
                i=j;
                x=x+1;
                break;
            end
            if j==signal4;
                break;
            end
        end
    else
        i=i+1;
    end
end

for z=1:size(index4,2)-1;
    if index4(z+1)-index4(z) > 22;
        begin=z+2;
        break;
    end
end

t2_i=begin;
while t2_i < size(index4,2),
    if index4(t2_i+1)-index4(t2_i) < 10;
        tag2(k)=0;
        t2_i=t2_i+2;
    elseif index4(t2_i+1)-index4(t2_i) >= 13;
        tag2(k)=1;
        t2_i=t2_i+1;
    else
        tag2(k)=2;
    end
    k=k+1;
end

reader1=num2str(reader1);
tag1=num2str(tag1);
reader2=num2str(reader2);
tag2=num2str(tag2);
disp('R->T:␣')
disp(reader1)
disp('T->R:␣')
disp(tag1)
disp('R->T:␣')
```

```
disp(reader2)
disp('T->R: ')
disp(tag2)
```