Final project: Key Finder

Embedded FW Essentials

June 2015

Sandeep Verma, VK Taank

Objective: To apply the learnings of the embedded class and create a project based on mbed board LPC1768.

Final Project: RF Key Finder

Project Details:

Using two mbed boards and a pair of RF transmitter and receiver, create a key finder. One mbed board is acting as a Transmitter (which is connected to the key) and the other board acts as a receiver and when activated will look for a specific signal from the transmitter. If the encoded signal matches with expected key then receiver mbed board will calculate the time taken between two messages to compute the rough distance between the transmitter (key) and receiver(key finder). When moving away, the reciver will buzz slower and when moving towards the receiver will buzz faster. A similar message will be printed on the crystal display to confirm if going towards the key or away from the key.

Limitations:

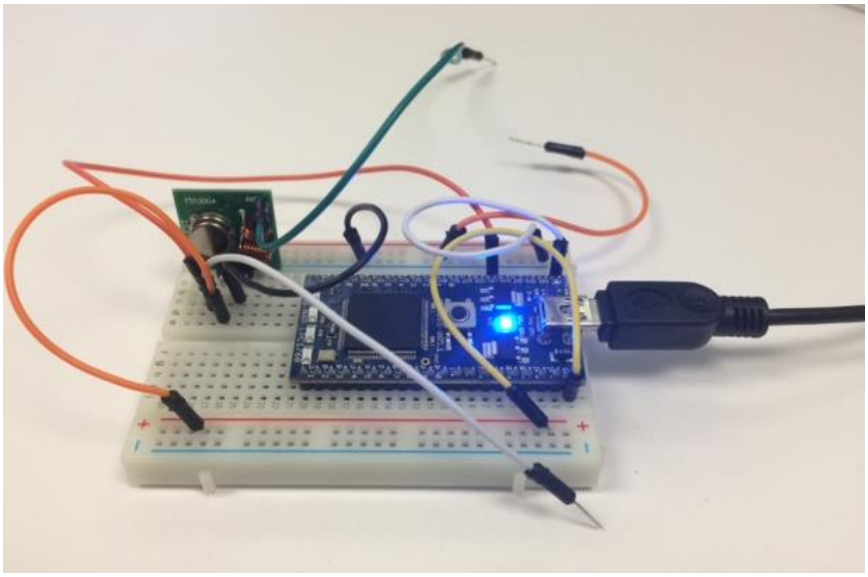
1. RF link is very basic and easy to implement link but it is not a suitable for measuring distance between receiver and transmitter. A SONAR could be used here but it will not work if there is an obstacle between receiver and transmitter. A RF link will work but the distance calculation might not be accurate.
2. The components used in this project are not very reliable and we see a lot of noise in the link. The software filtration does not work all the time.

Hardware Used:

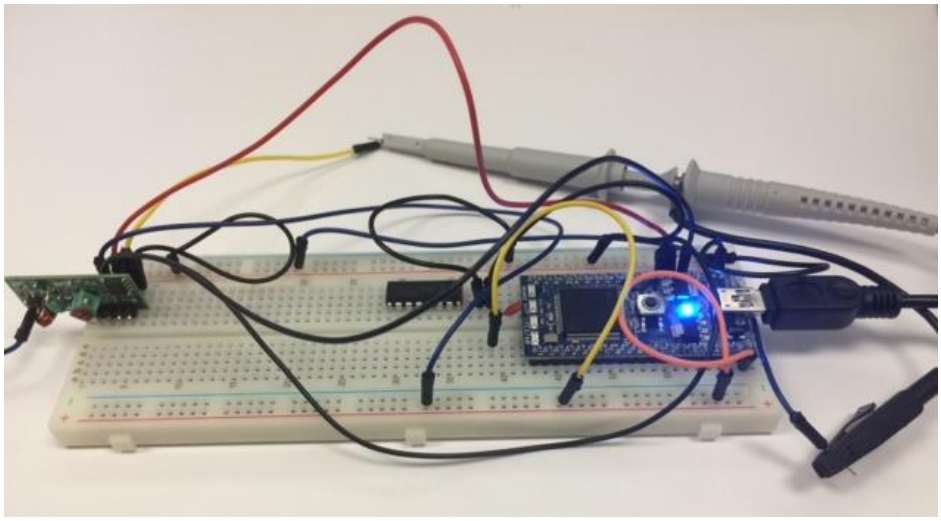
1. Two mbed boards (LPC 1768)
2. Two breadboards
3. One RF Transmitter (Model: XD-FST)
4. One RF Receiver (XD-RF-5V)
5. LCF Crystal Display
6. Sound Buzzer
7. Connecting cables

Project Setup:

1. Transmitter
    a. Use one mbed board as Transmitter
    b. Use Pin7 as Digital Out and Transmit data through Pin7
    c. Pin7 is connected to Data-in of RF transmitter



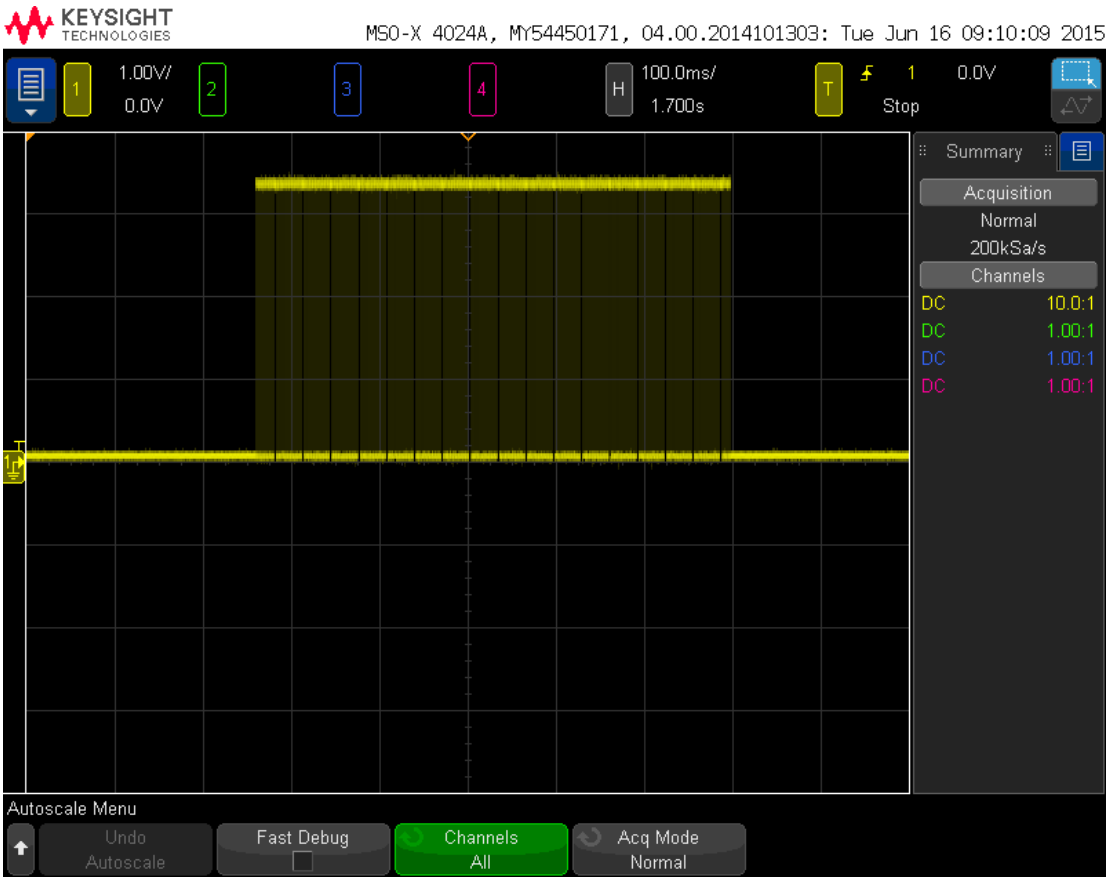
2. Receiver
    a. Use one mbed board as receiver
    b. Use pin5 as interrupt and is connected to Data pin1 of receiver
    c. Use pin7 as DigitalIn and is connected to Data pin2 of receiver
    d. Pin8 is connected to the Positive of the Sound Buzzer



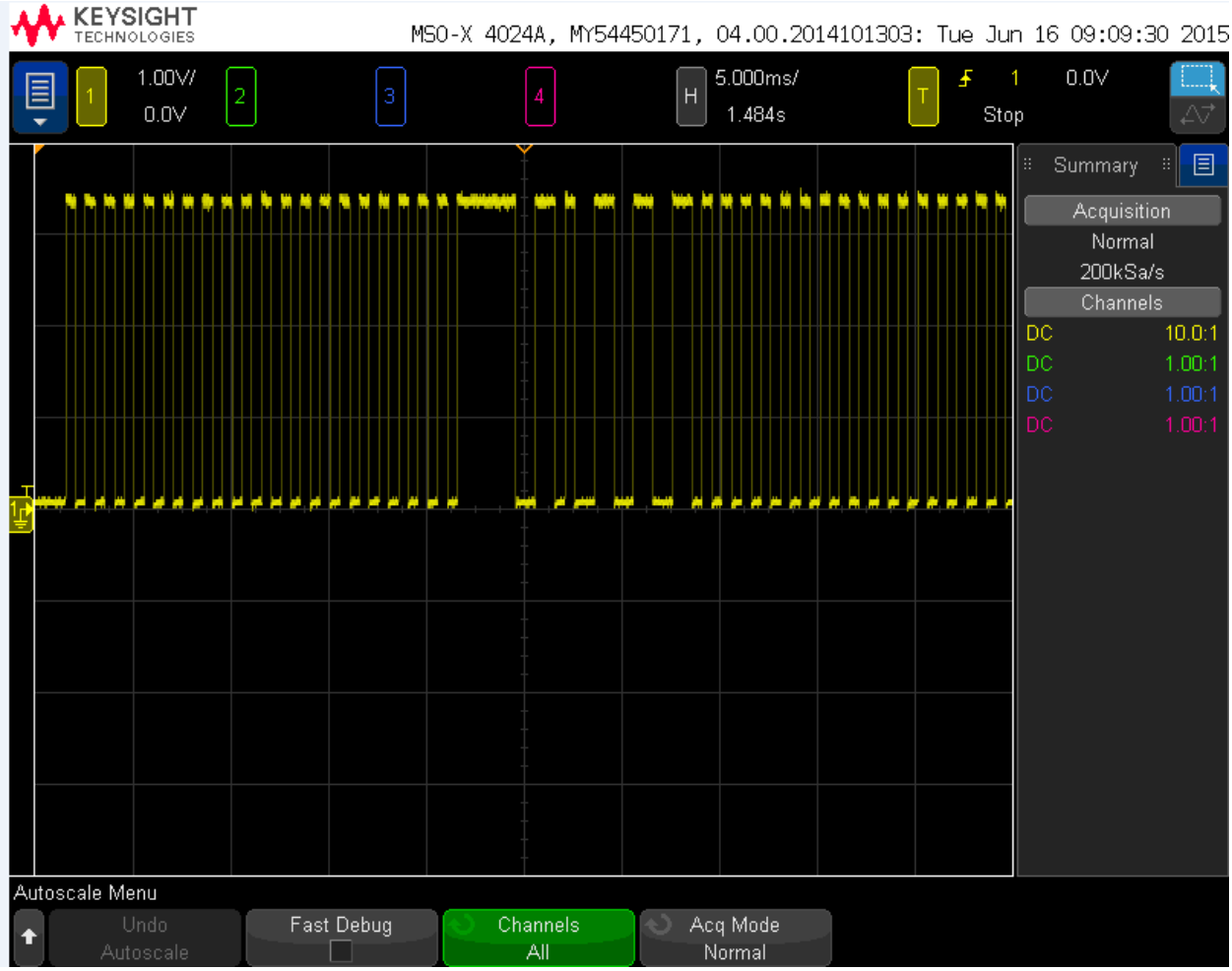
As there is a lot of noise on the link, we came up with the idea of software filtering. We created a protocol which the receiver reads and take action only when the send signal is decoded by the receiver.

Transmitter:

1. Send 20 pulses of 1ms cycle or 500us low and 500us high
2. Wait for 3ms
3. Send data bit and wait for 500us and send data compliment and wait for 500us (Data and Data bar is sent to make sure we can catch any bit stuck and confirm the data read is the right data and not any noise. The receiver expects the data and data bar. If the data bar is not present it will not consider the data as a correct data and restart the signal sensing again)
4. Force signal low
5. In project we send signal “VK TAANK SANDEEP#”
6. Scope capture attached below



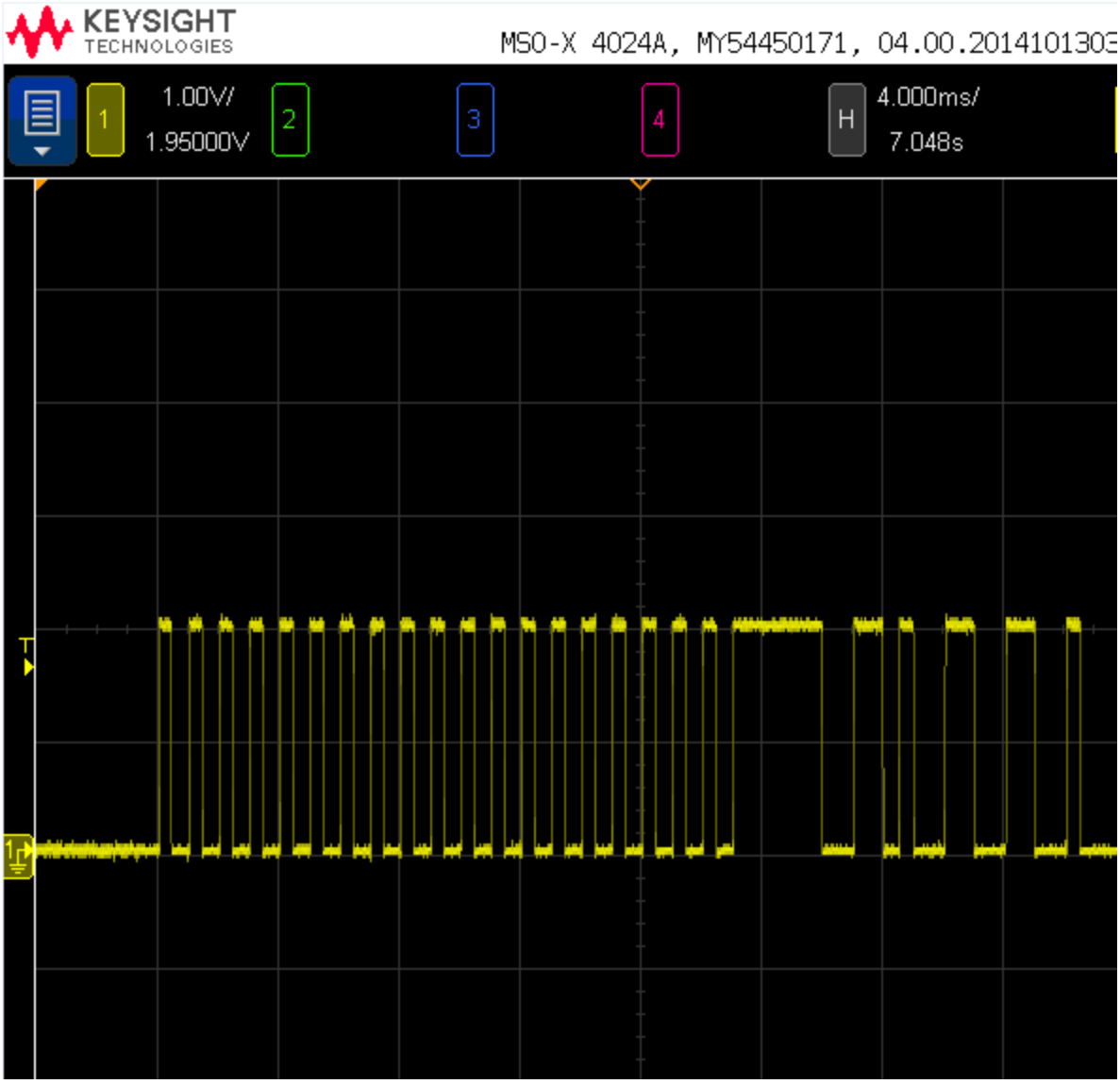
Pic3: Signal captured from Transmitter side at 100ms window. The line stays low and then send the pulse and data



Pic4: First20 pulses are sent, then a wait of 3ms and the the data transmitted here is " 0101 0110" which is hex 0x56 or ASCII "V". The process repeats till all the data is sent.

Receiver:

1. Set pin5 as interrupt
2. Wait for interrupt to go high (attach the sensing code to the rising edge of pin5)
3. Start counting the next 20 pulses (cycle of 1ms)
4. If the pulse count is 20 then it is a signal for receiver that the receiver is in the right snapshot window and is prepared to receive the actual data. If no match then sensing starts all over again till the interrupt goes high again
5. Once the 20pulse signal is sensed, mbed waits for 3ms and then starts reading the data through DigitalIn pin7 bit bit by bit and print every character to the Teraterm terminal (through USB serial function)



Pic5: Reciver recives 20 pulses, 3ms wait and data which here is " 0110 1010" which when read by the software as " 0101 0110" or 0x56 or ascii "V". Rest of the data is followed.

Work still left:

1. Due to a lot of noise the decoding is not working correctly. We will continue fine tuning the antenna length and our code.
2. The distance measurement is not done yet.
3. Display/sound part is not done yet.
4. DATA bar verification is not implemented yet.

Project Code: Transmitter

```
#include "mbed.h"



Serial pc(USBTX, USBRX); // tx, rx
//Serial uart(p13, p14);   // serial port tx at p13 and rx at p14
Serial out(p13, NC);
DigitalOut DataTx  (p6);


#define TAKE_N_BITS_FROM(b, p, n) ((b) >> (p)) & ((1 << (n)) - 1)


void Transmit(char input){
 int i;


 for(i=0; i<20; i++)
 {
  DataTx=1;
  wait(0.0005);
  DataTx=0;
  wait(0.0005);
 }


 DataTx=1;
 wait(0.003);
```

```
DataTx=0;
wait(0.0005);




for(i=0; i<8; i++){
if (TAKE_N_BITS_FROM(input,i,1)==1)
//if(bitread(input,(b.b1))==1)
{
DataTx=1;
}
else
{
DataTx=0;
}
wait(0.0005);

if (TAKE_N_BITS_FROM(input,i,1)==1)
//if(bitread(input,b.b1)==1)
{
DataTx=0;
}
else
{
DataTx=1;
```

```
  }
  wait(0.0005);
}

  DataTx=0;
}



int main(){



while(1){

  pc.printf(" START SEND ");
  Transmit('V');
  Transmit('K');
  Transmit(' ');
  Transmit('T');
  Transmit('A');
  Transmit('A');
  Transmit('N');
  Transmit('K');
  Transmit(' ');
  Transmit('S');
  Transmit('A');
```

```
  Transmit('N');
  Transmit('D');
  Transmit('E');
  Transmit('E');
  Transmit('P');
  Transmit('#');
  DataTx=0;
  wait(5);
  //v++;
  pc.printf(" FINSH SEND ");
}


}
```

Project Code: Receiver

```
#include "mbed.h"



Serial pc(USBTX, USBRX); // tx, rx

//Serial uart(p13, p14);   // serial port tx at p13 and rx at p14

//Serial out(p13, NC);

DigitalIn DataRx  (p7);

InterruptIn button(p5);



int i, good, good1, good2, count_total;

uint8_t data_in=0;

//char data_read;


class Counter {

public:

   Counter(PinName pin) : _interrupt(pin) {       // create the InterruptIn on the pin specified to Counter

      _interrupt.rise(this, &Counter::incrementhigh); // attach increment function of this counter instance

      _interrupt.fall(this, &Counter::incrementlow);

   }


   void incrementhigh() {

      _counthigh++;

   }
```

```
    void incrementlow() {
        _countlow++;
    }

    int readhigh() {
        return _counthigh;
    }

    int readlow(){
        return _countlow;
    }

private:
    InterruptIn _interrupt;
    volatile int _counthigh, _countlow;
};

Counter counter(p5);

int high,low;
void data_incoming(){

    pc.printf("Start the data sensing loop");
    for(i=0; i <40; i++){
```

```
   low=counter.readlow();
   high=counter.readhigh();
   wait(0.0005);
    }

pc.printf("Hightimes: %d LowTimes: %d\n",high,low);
if((high==20) && (low==20)){

            wait(0.0003);



            for(i=0; i<8; i++){
              uint8_t p=0x1;
              if(DataRx==1)
              {

                p=p<<i;
                //pc.printf("i=%i",i);
                data_in=data_in | p;
                //pc.printf ("data_in_insideloop %c",data_in);
                wait(0.001);
              }

              else
              {
```

```
            //pc.printf ("data_in_insideloop_and_elsestement %c",data_in);
            wait(0.001);
          }
        }

        char data_read=char(data_in);

        pc.printf("%c",data_read);

}

}

int main() {
   while(1) {
      button.rise(&data_incoming);
      //pc.printf("Count so far: Low:%d  High:%d\n", counter.readlow(), counter.readhigh());
      //wait(0.0005);
   }
}
```