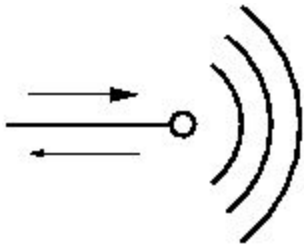
THE UNIVERSITY OF BRITISH COLUMBIA
Department of Electrical and Computer Engineering



MAJOR COURSE PROJECT

C.H.I.M.E Radio-telescope Transmitter

prepared by

Cyrus Cheung 77080844
Tonny Li 19313048

in Partial Fulfillment of the Requirements for

ELEC 411 - Antennas and Propagation

Date submitted: Fri, 20 Dec 2019

Abstract

Using the MAX41460 evaluation kit, we were tasked to: (1) transmit data with a desired frequency, (2) output a chirp signal as proposed by CHIME, (3) explore other potential signals that may be helpful to have on the radio calibrator. We have chosen the Arduino Due as our master device, and wrote a program based on the Arduino IDE which is able to control signal frequency and modulation, and send a discretized chirp signal. Using the evaluation kit as a starting point, we went about implementing our own program based off the software development kit provided and datasheet of the MAX41460. First of all, we set up the initial programming with the help of the MAX41460 datasheet. Then we want to reprogram the chip to transmit at our desired frequency. A function was written that enables us to reprogram the chip frequency and transmit and using this function we produced a chirp signal using a nested loop. To debug, we enabled the 4 wire SPI mode to verify we are writing registers correctly. We are also using a spectrum analyzer to ensure that the transmitted signal matches the desired signal. After testing, we have concluded that the MAX41460 is an easily programmable chip and this chip could potentially be useful as a radio-calibrator. If further signal processing such as FHSS and DSSS is desired, additional hardware is required, for which we have provided a template.

C.H.I.M.E Radio-telescope Transmitter

1 Objectives

Our objective for this project is to interface with the MAX41460 chip using our chosen micro-controller. Our first main objective is to transmit data with a desired frequency. Secondly, we want to output a chirp signal as proposed by CHIME. Our third objective is to explore other potential signals that may be helpful to have on the radio calibrator.

2 Approach

Using SPI to interface with the MAX41460 chip allows us flexibility with deciding our master device. For simplicity sake we've decided to use the Arduino Due , which is a 3.3V microcontroller and is compatible with our voltage requirement of 3.3V for our chip. We're using SPI to communicate to the MAX41460 chip and spectrum analyzer to view its output, our test setup is straightforward as shown:

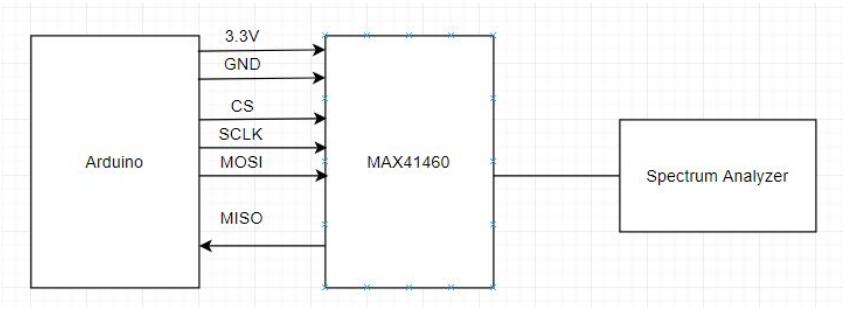


Figure 1. High-level block diagram of test system

To write to the registers in the MAX41460 chip, we decided to use software bit-banging. Since using arduino's hardware SPI inhibits you from setting MOSI, which is how you transmit logic 1's and 0's we must write our own code for software bit-banging.

For our first objective, In order to see if ensure we are correctly interfacing with our chip, we will enable 4-wire mode MAX41460's which turns the CLKOUT pin into a MISO pin used for debugging. As per instructions from the data sheet, on power-up of the chip we must burst write 17 registers, the chip will auto increment the address when you successfully write 8 bits into a register. Thus we will need 144 SCLK ((17 registers + 1 address) * 8) cycles to complete. There are specific values you must set during initial programming, as stated by the data sheet:

Initial Programming

After turning on power supply (or after a soft reset), an SPI transaction that burst-writes 17 consecutive registers from address 0x00 to 0x10 is required to initialize the PLL frequency synthesizer.

The initial programming must clear MODMODE (register CFG1, address 0x00, bit 0), clear SPI_TXEN1 (register CFG6, address 0x0A, bit 1), configure FREQ[23:0] (register PLL3, PLL4 and PLL5) to desired frequency, and set SPI_TXEN2 (register CFG7, address 0x10, bit1).

Figure 2. Instructions for Initial Programming

After initial programming, we can set MODMODE, frequency, and start transmitting by setting SPI_TXEN2 to 1, enabling transmission mode. To transmit a logic 1, we set MOSI to HIGH, and to transmit a logic 0, we set MOSI to LOW. At this point we will use a spectrum analyzer to ensure that the chip is transmitting. For configurations options, please refer to the register map in the datasheet. In order to select the frequency at which to transmit, we will write a function that will input a frequency, convert it into the correct value, then write to the 3 registers associated with frequency and transmit.

Our second objective is to attempt to produce a chirp signal. Once we can easily change frequency as required from objective one, producing a chirp signal should be effortless. This chip does not allow a changing of frequency during transmission mode, so we must turn off transmission, reprogram the frequency, then re-enter transmission mode. This process takes 48 SCLK cycles (address 0x0B to 0x10) and an additional 10us. If our data transmission rate is very slow (kbps), then the time to change frequency is negligible. But it also means that our chirp signal will be discretized. This chirp function will call the set frequency function in nested loop to provide the chirp.

For our third objective, the signals we've explored are: direct sequence spread spectrum (DSSS) and frequency hopping spread spectrum (FHSS). Luckily for us, a DSSS and FHSS signal employ the same hardware. A DSSS signal is a wider band signal compared to the FHSS. The idea behind DSSS modulation is that we can spread our signal such that it is low power spread across a very wide bandwidth as shown:

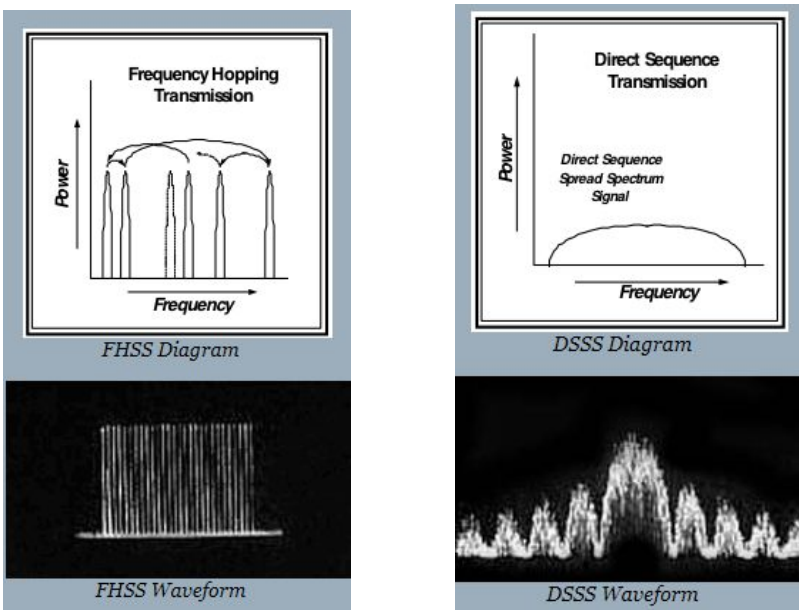


Figure 3. DSSS AND FHSS signal example from https://cs.stanford.edu/people/eroberts/courses/soco/projects/2003-04/wireless-computing/int_ss.shtml

A DSSS signal is resilient to an impulse interference at a narrow range of frequencies. This signal also appears as noise to a third party observer. The disadvantage of the DSSS signal is you have have very few users using this technique since it occupies a large range of frequencies.

For a FHSS modulation, the transmitter sends a signal based on a PN sequence, and the receiver follows the jump using the same PN sequence. Data is only transmitted in small time period before jumping onto another frequency. The FHSS is shown above in figure 3. The advantage of this method of modulation is that only the devices that know the pattern can interpret the signal, otherwise it's difficult to decipher. The FHSS modulation also offers resistance to interference. An impulse interference signal will only affect a small percentage of bits that are transmitted, and unlike the DSSS signal, we do not require such a large frequency range, which means there can be a lot of users within a frequency band. Bluetooth is a notable example of a technology that employs this method of modulation.

We cannot produce these two types of signal efficiently without additional hardware. To produce these signals we require a PN noise generator, and an RF mixer configured as shown:

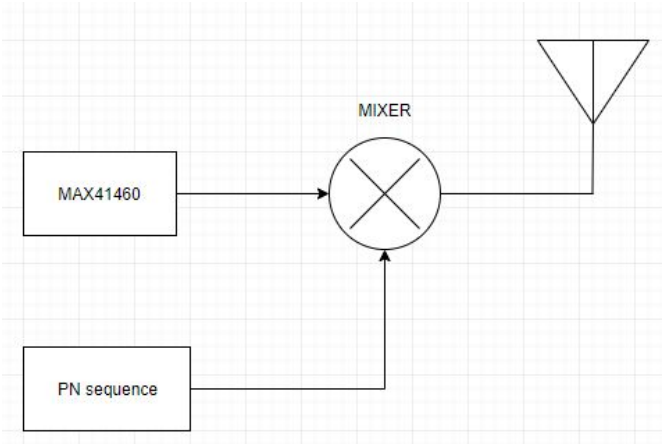


Figure 4. High level block diagram of components needed

To produce a DSSS signal we require a PN sequence clock rate that is much faster than the data transmission rate. As an example we transmitted data at 5kbps, and we can provide a PWM from the microcontroller at 1MHz. And to produce a FHSS signal we require PN sequence but at a slower rate, which the DUE can also provide.

For the PN sequence, we imagined using an LFSR controlled by a clock source from the onboard computer. We decided to add a MUX that allows the input of a shift register to be either be the feedback of XNOR gates of the LFSR, or the microcontroller so that we could set the seed data. This LFSR with MUX input will then be fed to our mixer. We selected our shift register to be 8 bits meaning our sequence has 2^8 - 1 bits. Due to time constraints, we arbitrarily chosen this number.

For the mixer we briefly explored using the MAX2660 which is a upconverter mixer that operates at 400MHz to 2.5GHz. This mixer will have attenuator pads on both the input and output to reduce reflection power back to our transmitter chip as a safety feature. All in all, our design would require 11 pins from the micro-controller. We will provide a mock up for this design using KiCAD.

Also note that, we technically do not require a mixer to produce a FHSS signal, to do it with software is infeasible and would take too much resources from the onboard computer.

3 Test Set-ups

The equipment that we used in this project includes a spectrum analyzer, an Arduino DUE board and MAX41460 evaluation kit. Our simple set-up is as follows:

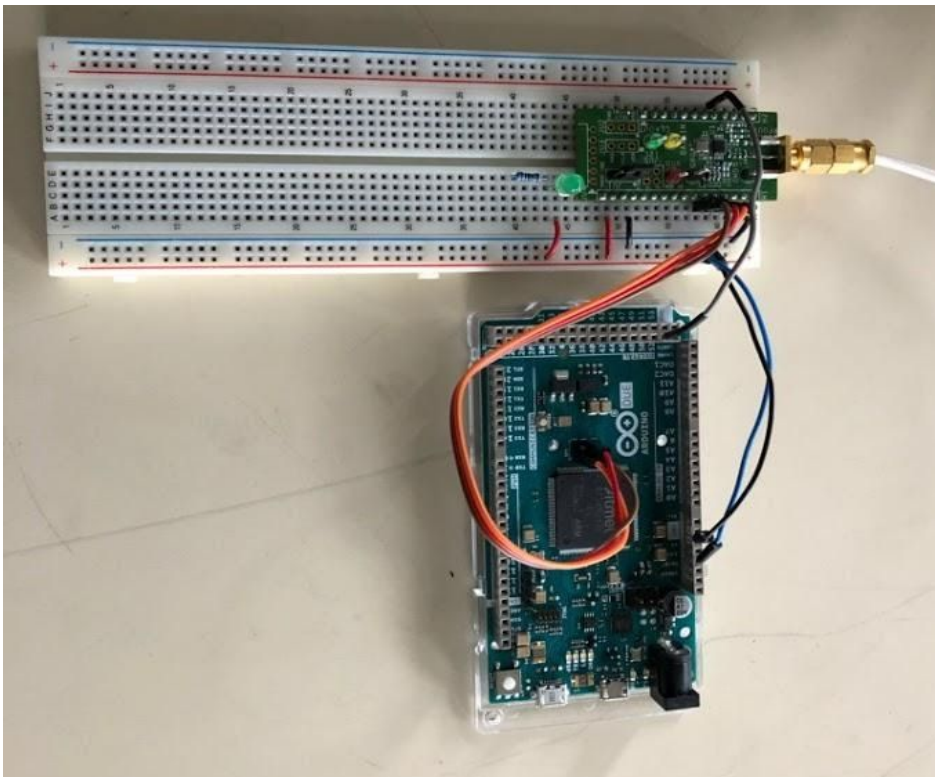
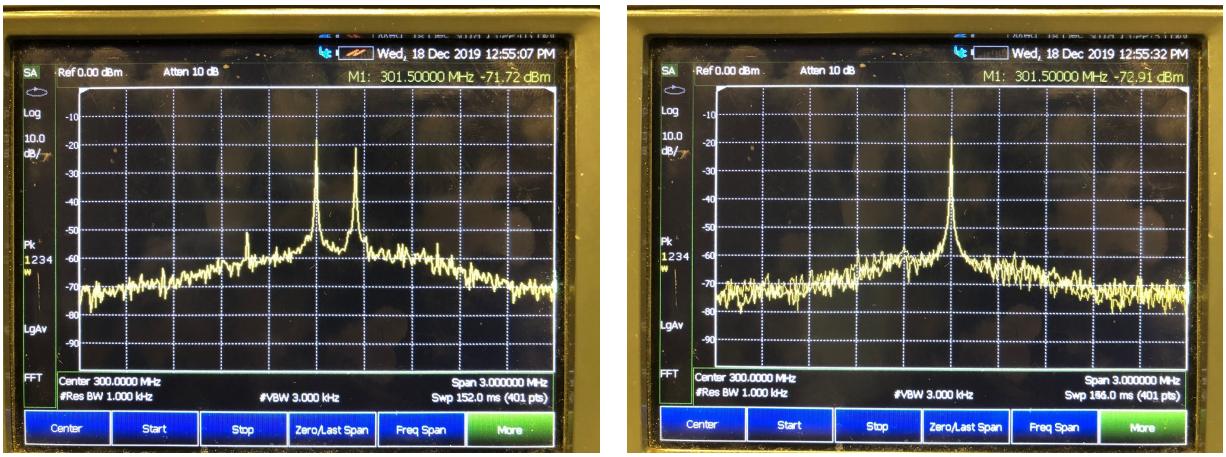


Figure 5. Picture of our set up, Due with the EV kit

The code written with platformIO IDE will be attached at the end.

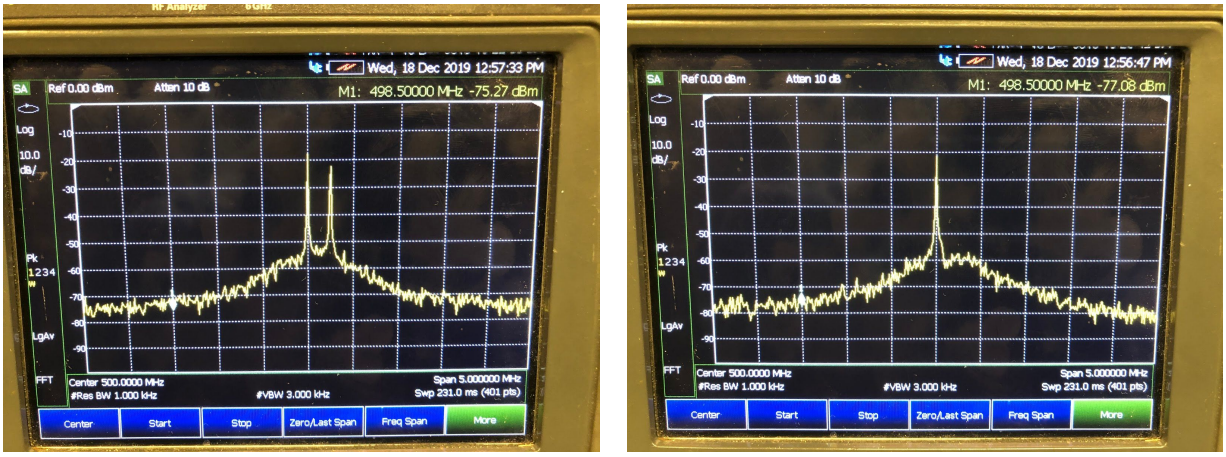
4 Results

For the first objective, the chip has the ability of setting up certain frequency based on a desired frequency, we set the frequency at 300MHz and 500Mhz using FM and AM as shown below in Figure 8 & 9.



FM Signal AM Signal

Figure 6. Tested Signal Using Spectrum Analyzer(300 MHz)



FM Signal AM Signal

Figure 7. Tested Signal Using Spectrum Analyzer(500 MHz)

For the second objective we had to produce a chirp signal. Because we did not have the equipment to measure a time domain signal of the chirp signal, we tried to measure our chirp

signal using the spectrum analyzer. Our chirp signal is played on loop from 400-415MHz with increments of +1 MHz. Our chirp signal is shown in figure 7:

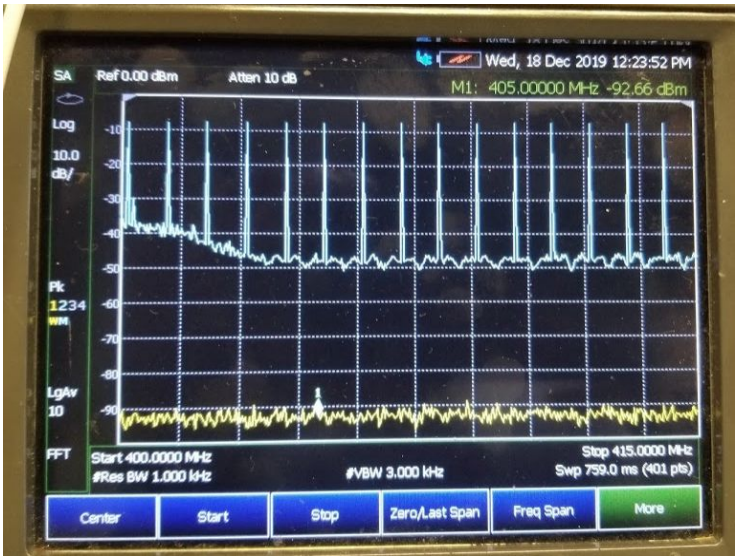


Figure 8. Discretized chirp signal

If a chirp signal with smaller increments or larger frequency range is desired, then it is easy to modify using our code.

For our template design of the MAX41460 with RF mixer and PN generator, we would require the following chips:

Description	Part number	Quantity
Transmitter	MAX41460	1
Mixer	MAX2660EUT+T	1
Shift Register	SN74HC164PWR	1
XNOR Gate	SN74HC266DR	1
MUX	SN74HC153NG4	1

Table 1. Chips recommended for DSSS/FHSS implementation

Note that, these parts require passive components that we did not list. Our additional hardware will require 5 additional pins to control the PN-generator.

Pins	Pin name	Pin description
1	MCU 3.3V	voltage supply
2	MCU GND	Ground
3	SCLK	clock for MAX41460
4	MOSI	data in for MAX41460
5	MISO	MAX41460 output data
6	SS	select
7	CLR	clears shift registers
8	SEL1	sel for MUX to shift register
9	DATA	data in to MUX
10	CLK	clock for LFSR
11	EN	enable for shift register

For the schematic of our MAX41460 chip with an RF mixer and PN generator please refer to appendix.

5 Conclusions and Recommendations

As discussed in the objectives, we were able to start transmission and produce a chirp signal. We also discussed the possibility of producing DSSS and FHSS signal. With the MAX41460, we can easily achieve objective 1 and 2. To start the chip, we must do a start-up initial programming. Then if we want to change the transmission parameters, we simply complete an SPI transaction then set SPI_TXEN2 to 1 to re-enable transmission with a new parameters. Although the chirp signal is not continuous, the reprogramming time very quick (10us), that it should not be a problem.

As for the DSSS and FHSS we require additional hardware that is not equipped with the MAX41460 evaluation kit. Our recommended implementation would require 5 additional pins from the microcontroller. Of course due to time limitations, this is a rough idea of what the final design should be, and should be used as a framework for a more complete design.

Moving forward, we recommend exploring the feasibility of producing a FHSS and DSSS signal.

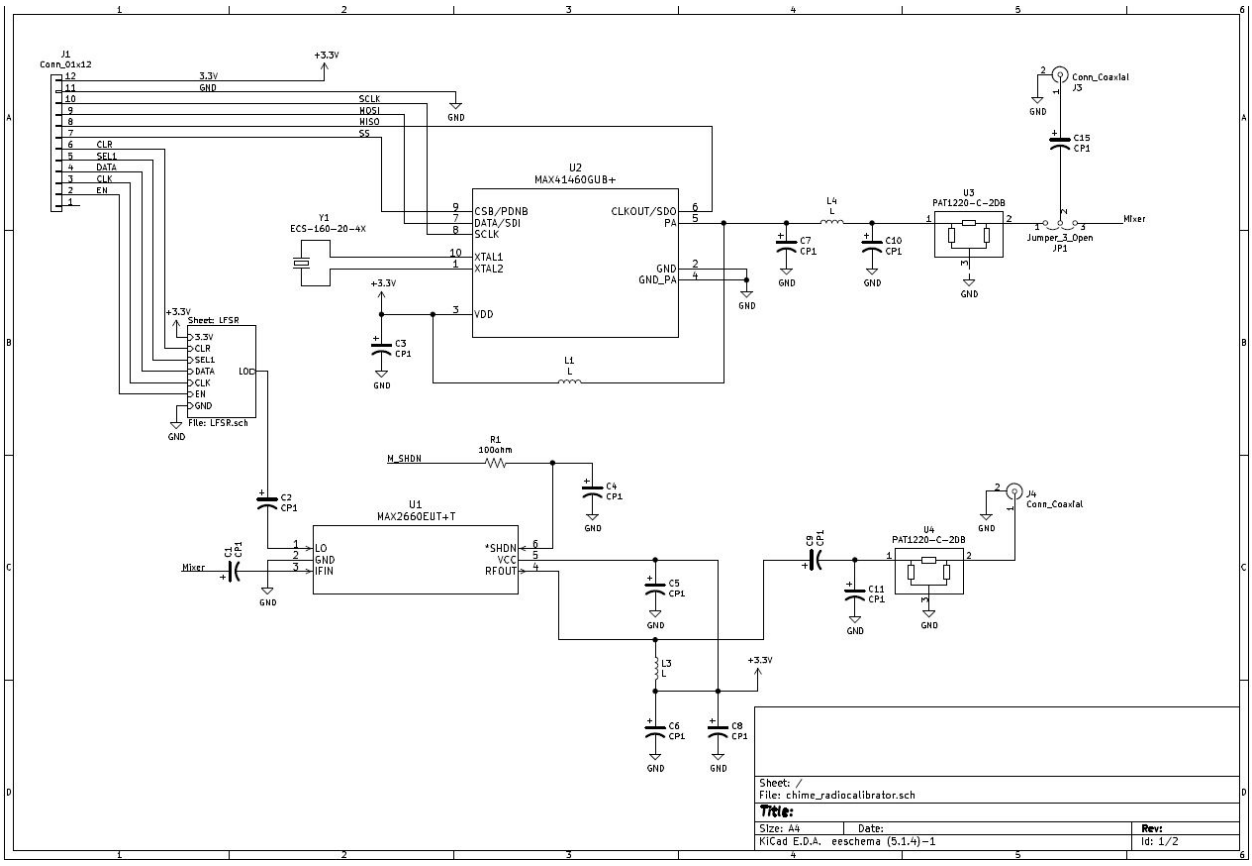
REFERENCES

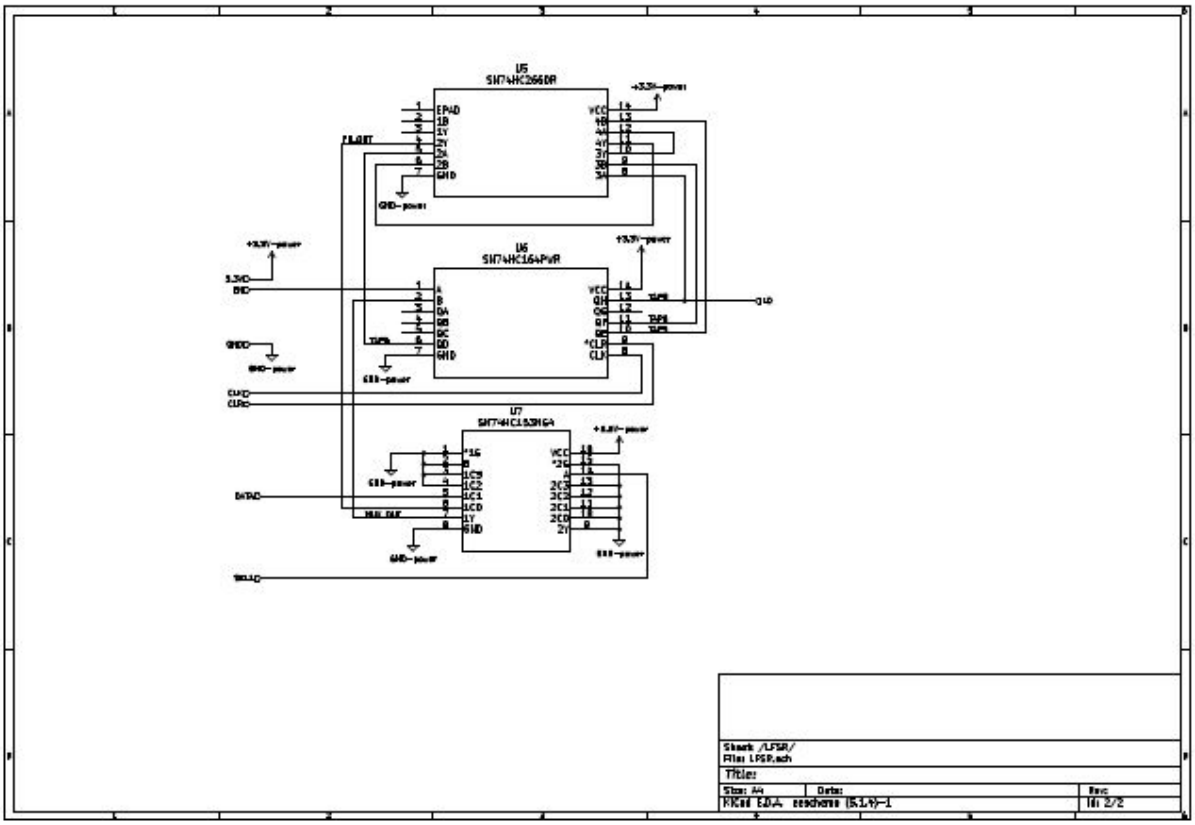
[1]https://cs.stanford.edu/people/eroberts/courses/soco/projects/2003-04/wireless-computing/int_ss.shtml

[2] https://datasheets.maximintegrated.com/en/ds/MAX41460.pdf

[3]https://datasheets.maximintegrated.com/en/ds/MAX41464EVKIT-MAX41464EVKIT-868.pdf

APPENDIX I


J1
Conn_01x12
3.3V
GND
SCLK
MOSI
MISO
SS
CLR
SELL
DATA
CLK
EN
U2
MAX41460GUB+
CSB/PDNB
DATA/SDI
SCLK
XTAL1
XTAL2
VDD
CLKOUT/SDO
PA
GND
GND_PA
Y1
ECS-160-20-4X
L4
U3
PAT1220-C-2DB
J3
Conn_Coaxial
C15
CP1
Mixer
Jumper_3_Open
JP1
C7
C10
C3
L1
Sheet: LFSR
File: LFSR.sch
LOC
R1
100ohm
M_SHDN
U1
MAX2660EUT+T
LO
IFIN
*SHDN
VCC
RFOUT
C2
C4
C1
C5
C9
C11
U4
J4
L3
C6
C8
Sheet: /
File: chime_radiocalibrator.sch
Title:
Size: A4
Date:
Rev:
KiCad E.D.A. eeschema (5.1.4)-1
Id: 1/2


U5
SN74HC266DR
U6
SN74HC164PWR
U7
Sheet /LFSR/
File: LFSR.sch
Title:
Size: A4
Date:
Rev:
KiCad E.D.A. eeschema (5.1.4)-1
Id: 2/2