EE 100/42

Project: Infrared Receiver Design

version 2.0

Zhiming Deng and A. M. Niknejad
Berkeley Wireless Research Center
University of California, Berkeley 2108 Allston Way, Suite 200
Berkeley, CA 94704-1302

April 11, 2010

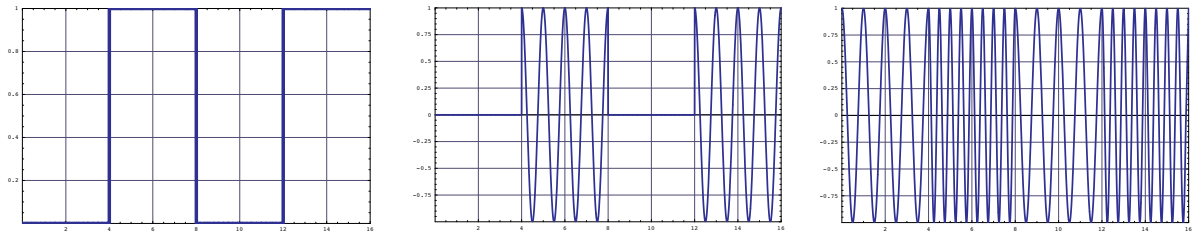
1 Introduction

In this project you will be designing a wireless infrared receiver. This project builds on the various laboratory mini-projects that you have done up to now, including the design of the light detector and the microcontroller interface design. The system works using an infrared light source (LED) and a light sensor (photo-transistor). The binary transmitted data stream is used to modulate the emitted light level. The receiver should work under varying input signal levels, which corresponds to varying impinging infrared light levels, and should be able detect the transmitted information regardless of the of the levels of the zero and one signals (as long as sufficient signal-to-noise ratio is sufficient). The detected bits should be read into the microcontroller and then subsequently sent out for display on the oscilloscope (or displayed on an external LCD) to demonstrate successful transmission.

2 Background

Wireless information transmission using electromagnetic radiation is ubiquitous. Depending on the requirements such as range (distance) and throughput (the rate at which information is transported), various frequency ranges are selected. RF frequencies, from kHz (1000 Hz) to tens of gigahertz are commonly used in mobile phones (600 MHz - 2 GHz) and wireless LAN (WLAN, WiFi) 2.4 GHz systems. Transmission at a particular frequency is strictly regulated, licensed and managed by the FCC to minimize interference and to protect critical communication to take place in emergency bands, military bands, radar, GPS. Operation in certain unlicensed frequency ranges, such as the the Industrial Scientific Medical (ISM) band, is regulated only in terms of maximum allowed radiation into the band. This is why operation in these bands has become so popular. Cordless phones often operate in the 900 MHz or 2.4 GHz ISM bands for this reason. Likewise, most commercial radios such as Bluetooth, Zigbee, and WLAN operate at 2.4 GHz. Operation at higher frequencies, such as in the 60 GHz ISM band, requires high frequency transistors and communication is essentially line of site and very short range. Communication using optical or infrared frequencies is also possible, but instead of designing antennas (which would have to have microscopic scales at these frequencies) we employ light emitting diodes (LED) or lasers. The infrared frequency range is a popular choice for remote controls and short range communication links. It's low cost, has a small profile and is less immune to interference from visible light. Unfortunately, it is limited to line of sight (LOS) applications.

Notice that in general the transmission frequency (say 900 MHz) is different than the rate at which information changes (say 20 kHz). The rate at which information changes is quantified as the bandwidth of the signal. For instance, a voice and music signals are contained approximately in the range 100 Hz to 20 kHz. But this signal cannot be transmitted directly (an efficient antenna would be too large, about 10 km) and therefore it must be first up-converted to a carrier frequency. This process is called "modulation" since the properties of the RF carrier signal is modulated by the information (phase, frequency, and/or amplitude). The simplest kind of modulation is amplitude modulation, which in its simplest form is on-off keying (OOK), wherein the carrier is essentially turned on and off to convey a 0 and a 1, as shown in Fig. **??**. Frequency modulation is also shown, where the carrier frequency



(a) (b) (c)

Figure 1: (a) The data waveform is used to modulate the (b) amplitude of the carrier or the (c) frequency of the carrier.

is modulated from one frequency to another to denote "1" or "0". If the current in an LED diode is modulated, the intensity of light is also modulated and this is the method by which information is transmitted in your project.

To pick up an optical signal in your receiver, you will employ a phototransistor, a device that can be modeled as a current source with varying current dependent on the impinging light intensity. Your circuit will sense the light level using this sensor and then convert it into a digital signal.

3 Data Packets, Headers, and Information Coding

Often times the raw stream of data is broken up into packets and and the information bits are compressed and encoded prior to transmission. Packetizing data is convenient because each packet contains a header which is used to identify the packet (used for error detection and retransmision protocols), it aids the synchronization process between the transmitter and the receiver and it helps to determine the amplitude of the received signal. In a wireless environment the received signal strength (RSS) varies because the distance between the transmitter and receiver is variable. Most systems employ a variable gain amplifier (VGA) to condition the signal so that it has the optimal amplitude for detection. By examining the header of the packet at the receiver, not only do we know the average level of the signal received, but we also can determine the level for a 0 and 1. Since the header is fixed and contains a known pattern, the levels for the 0 and 1 can be determined for each packet. The header is also very important for synchronization of the transmitter and receiver. Synchronization is necessary since the transmitter clock and the receiver clock are never perfectly synchronized, as shown in Fig. 2. Both systems must of course agree to a fixed clock rate for the data. A random timing offset between the two results in errors if the data is sampled at the wrong time (during the transitions). In addition to a phase offset between the clocks, the frequencies can be slightly different, which can be modeled as a time-varying phase offset. Finally, there is also a random drift (jitter) due to the imperfections and noise in the clocks.

Encoding introduces redundancies in the data that can be used to determine if transmission errors occurred. For instance, a simple parity check bit can be added to a packet based on the number of ones transmitted (odd or even). This will catch a single bit error but not two errors (or in general an even number of errors). In this project we will not employ coding

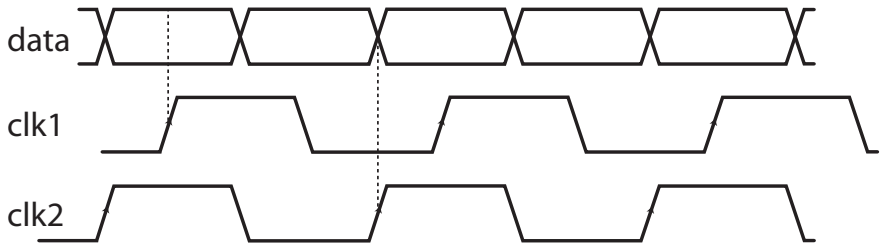


Figure 2: The importance of clock synchronization is illustrated. The first clock is synchronized and so data is sampled at the optimal point. In the second case data is sampled at the worst point, during transitions, which results in errors.

but any real communication link includes coding. In a real system, when a packet error is discovered, the system would ask for that garbled packet to be retransmitted. The header of each packet contains some identification (packet #52) and this information is sent back to the transmitter to request a retransmission. The packet length is chosen to balance the trade-off between packet header overhead and the Packet Error Rate (PER) of the system. Longer packets obviously incur less overhead but are more likely to be received in error.

Errors in transmission occur due to noise and interference. Noise also determines the maximum range for the communication link since transmitting a 1 must obviously be bigger than a 0, which will have an RMS amplitude determined by the noise level in the system. Noise is picked up by the detector (or antenna) from various sources, including the electronics themselves and through other natural sources (all bodies not at absolute zero radiate electromagnetic energy to maintain thermal equilibrium).

4 Transmitter Design

This sections describes the transmitted infrared signal that you must detect. The transmitter is predesigned and the characteristics of the signal and the header format are specified.

4.1 Packet Structure

In this project the packet contains 12 bytes. The first three bytes are the header and the rest of the byes are the payload, or the transmitted data bits. Each byte of data corresponds to an ASCII code (a = 0x61 or 01100001). The order of the bits is from MSB to LSB. The header contains the sequence of 0xFF (11111111) twice and then a sequence of zeros, 0x00 (00000000). The next 9 bytes are the message to be displayed. In this project, the same packet will be resent repeatedly which allows us to ignore synchronization. In other words, periodically the transmitter and receiver clocks will be out of phase and errors in the transmissions will occur. But as the clock phases realign, we can correctly detect the data. In a real system synchronization is done by either oversampling or by recreating the transmitters clock from the data (clock and data recovery (CDR)).

4.2 Transmitter Architecture

Fig.3 shows a block diagram of the transmitter. A digital sequence is generated by the MSP430 and sent into an amplitude and offset control block. The output is sent to the IR TX (Infrared Transmitter). The flow of the program the digital transmitter architecture is shown in Fig. 4. The MSP430 is first setup so that it is interrupted every microsecond. The source of the interrupt is the ADC10 sampling, which is driven by the internal on-chip DCO running at 8MHz / 8 = 1 MHz. The data rate of the transmitted bits is 1.6 kbps. The output port is P1.0. The program simply outputs the header sequence 0xFF and then 0x00 and then the 9 bytes of data.

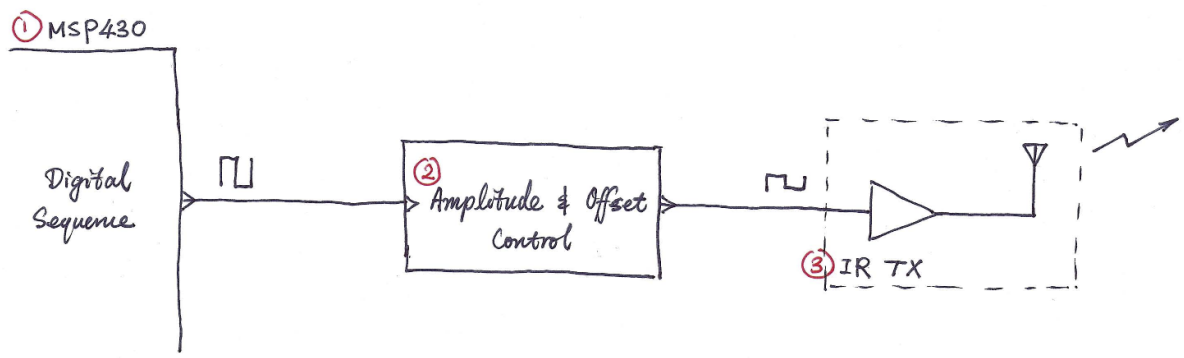


Figure 3: Block diagram of the transmitter

The amplitude and offset control circuitry, shown in Fig. 5, consists of a variable voltage divider, implemented with a potentiometer and a variable offset, provide by a DC voltage source. In this manner we can control the "high" and "low" output signal levels, which will be used to test the robustness of your receiver. Fig. 5 shows the HDSP-1000 IR transmitter, which contains an IR LED and a lens with sufficient power to transmit from centimeters to about 1 meter of distance. Since the output emission is in the infrared range, you will not be able to see the LED blink. On the other hand, if you view the LED through a digital camera (such as the ones found in your mobile phones), you will see the a bright infrared signal. This part draws a large current and should be carefully biased through a separate power supply.

5 Receiver Design

A block diagram of the receiver is shown in Fig. 6. The IR signal is received by the phototransistor sensor, amplified, and then digitized by a 1-bit analog-to-digital converter (essentially a comparator). While you could in practice use the built-in ADC to do most of the signal processing, we require that you build your own analog front-end. To aid the detection of the signal, though, we allow you to sense the header signal levels using the on-board ADC

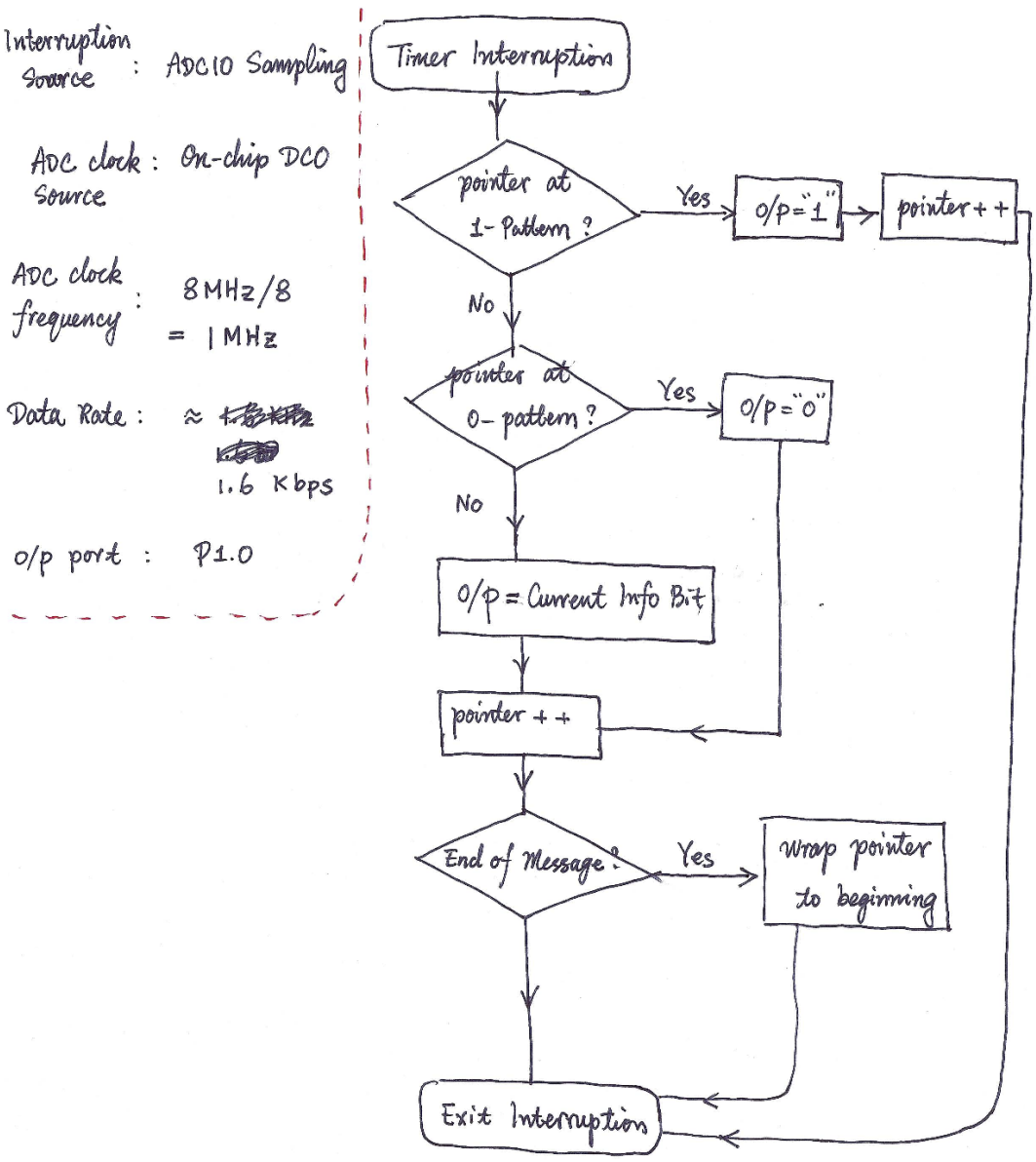


Figure 4: The transmitter program.

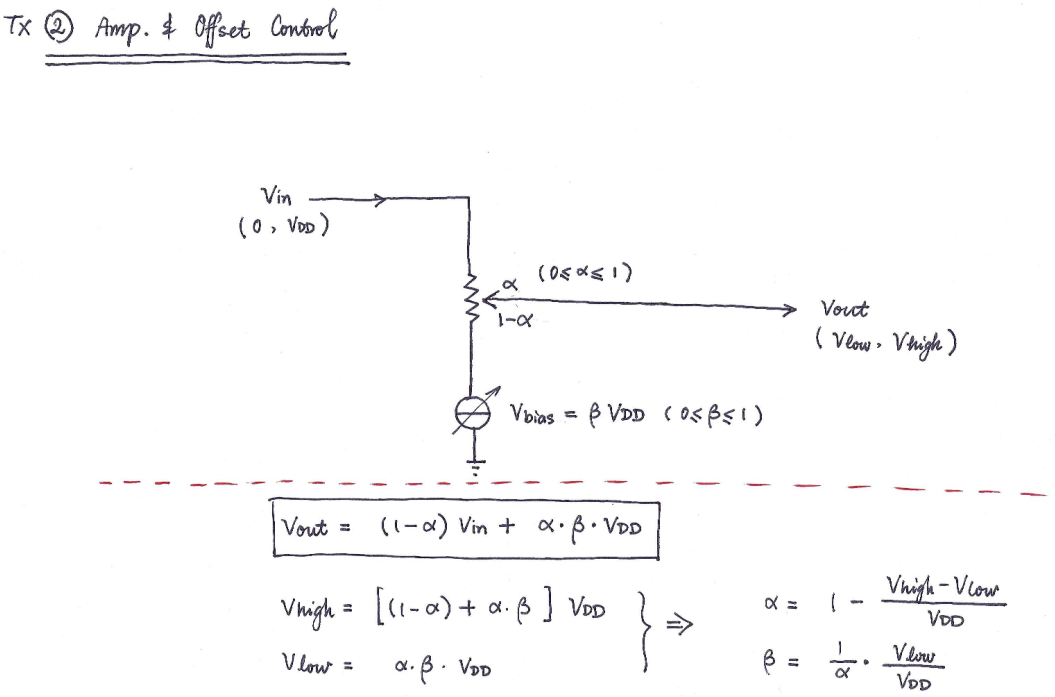


Figure 5: The amplitude and offset control blocks.

but once the signal level has been properly detected, digital control signals from the microcontroller are used to set the thresholds of the 1-bit ADC. In practice this mixed signal approach is useful when the data rate of the signal transitions exceeds the maximum data rate of the microcontroller ADC.

Fig. **??** shows the photoresistor, which can be modeled as a current source. This signal needs to be converted into a voltage and then buffered. Next this signal is digitized by using a circuit that compares the current input level to the threshold and decides whether to pass a "1" or "0" to the microcontroller. A comparator is a handy circuit to realize this functionality. To set the comparator threshold, you will need to generate an analog voltage from the microcontroller through a custom designed digital-to-analog converter (DAC). A binary weighted resistor ladder is a simple way to do this.

5.1 The Receiver ISR

The process flow for the receiver interrupt service routine (ISR) is shown in Fig. 7. The receiver microcontroller is setup in the following configuration. The ADC clock frequency is 4 MHz, which is 4 times faster than the transmitter clock. The analog input port is P1.85 and the digital input port is P1.1. The threshold control is output using ports P1.6, P1.5, P1.4,P1.3 (4-bits), where P1.6 is the MSB. The gain control is set using P1.2 and the serial display is output through P1.0.

There are essentially two states in the system, the analog sampling process and the digital sampling process. When we receive a new bit, we must first determine if were in the header or

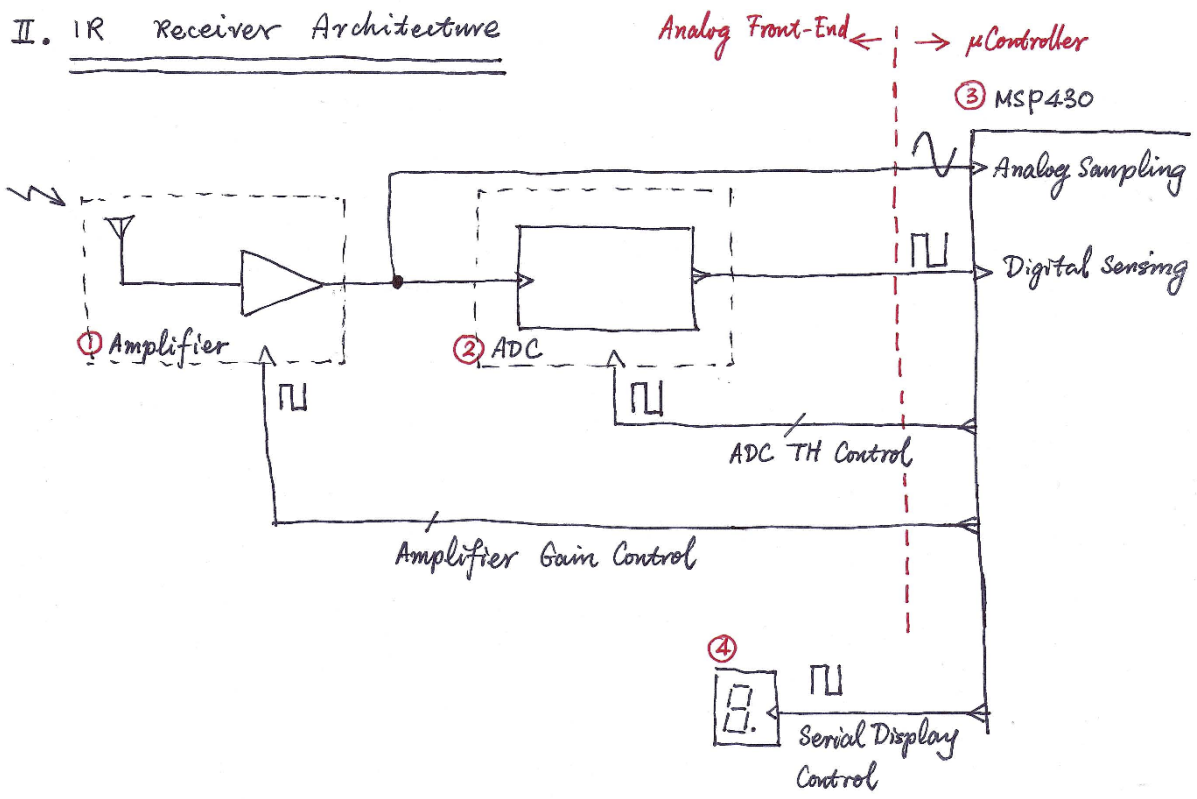


Figure 6: Block diagram of the receiver.

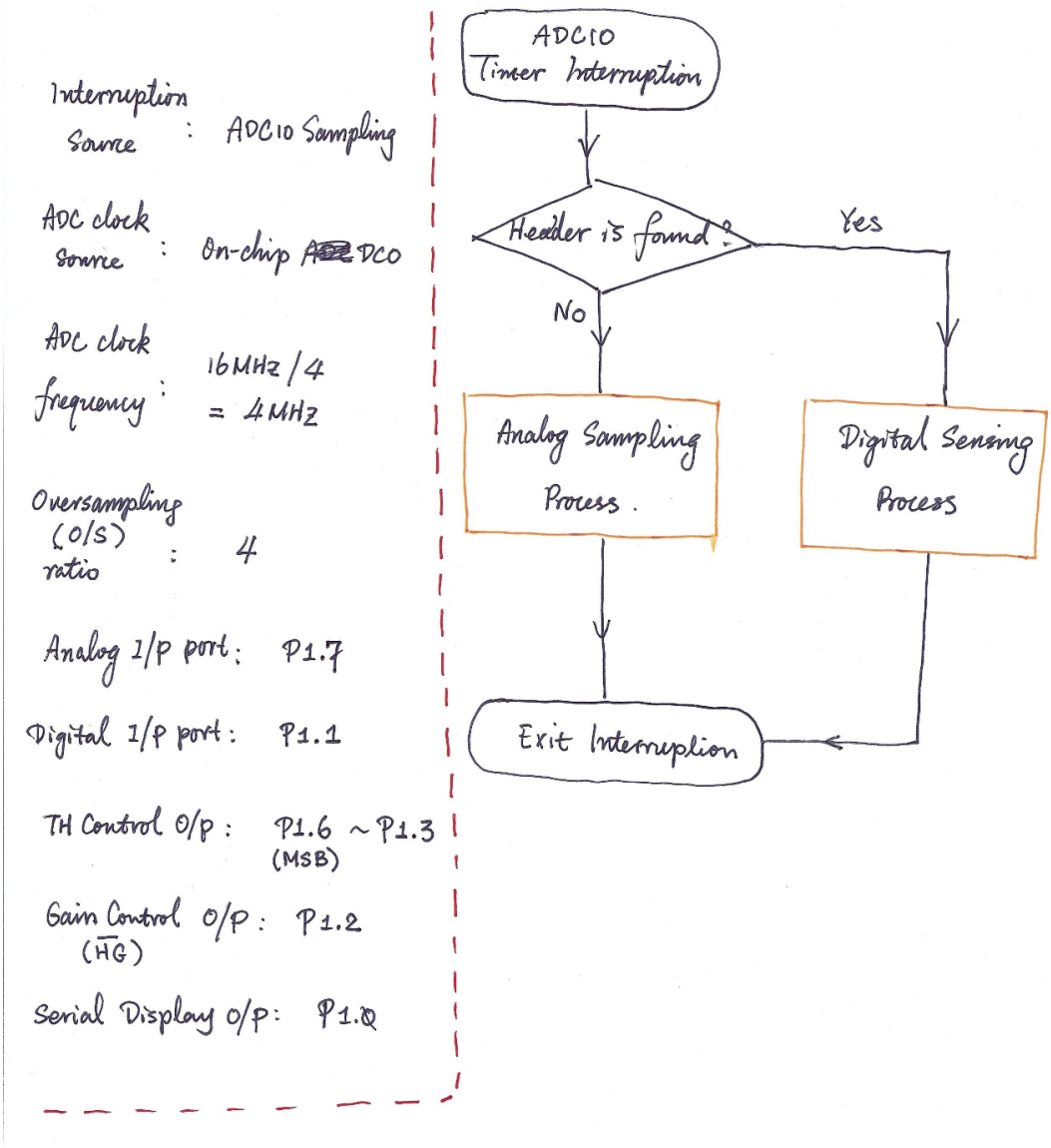


Figure 7: The receiver ISR.

in the data portion of the packet. During the processing of the header, the analog sampling process is used. Once the header has been read, the threshold voltage of the ADC is set. The program switches to the digital sampling process and reads in the digital samples provided by the ADC to the digital input of the microcontroller.

5.2 The Analog Sampling Process

During the analog sampling process, shown in Fig. 8, the data is oversampled by a factor of 4. When a new sample arrives, we store the results and then we first determine if a complete bit has been sampled. If not, we exit the routine. Once a complete bit has been sampled (4 clock periods), we calculate the variance of the sample. If the variance is larger than a certain threshold, the bit value varied substantially during the sampling process, which means that we are probably out of sync with the transmitter (the sampling clock is early or late). In this case we just drop the data and move on. If the data is sampled correctly, we then process the header. First we process the "1" values in the pattern-1 process to determine the level of the high input signal. Next we process the "0" values in the pattern-0 process to determine the low signal level.

The pattern-1 process is shown in Fig. 9. The routine first determines if the current bit is the same as the previous bit, which must be true during the "1" portion of the header. If we have not reached the end of the header (the counter saturates at the end), we calculate the "1" signal level by averaging. We then exit the process. During the pattern-0 process, shown in Fig. 10, we do the same thing to calculate the "0" signal level. Then this information is used to output the ADC threshold level and the VGA gain level.

5.3 The Digital Sampling Process

Fig. 11 shows the relatively straightforward digital sampling process, where each value is stored, and then after the completion of one data bit cycle, the value is decided and then the digital signal is output and displayed in a serial fashion (which can be used to drive an external display).

6 Design Project

Your project is to design the analog front end described above to detect the signal, amplify it, and then to digitize the signal using a 1-bit ADC. Your design should be robust and work with varying amplitude levels (corresponds to changing the distance). The GSI will adjust the transmitted amplitude and level of the transmitter to emulate the varying channel conditions. Since you are not performing synchronization in your circuit, your receiver may sometimes detect incorrect bits. You should detect this condition and avoid outputting wrong data. When the circuits resynchronize (due to drift), the correct output should be displayed. Since the same packet is continuously retransmitted, there is a lot of margin to allow for synchronization to occur.

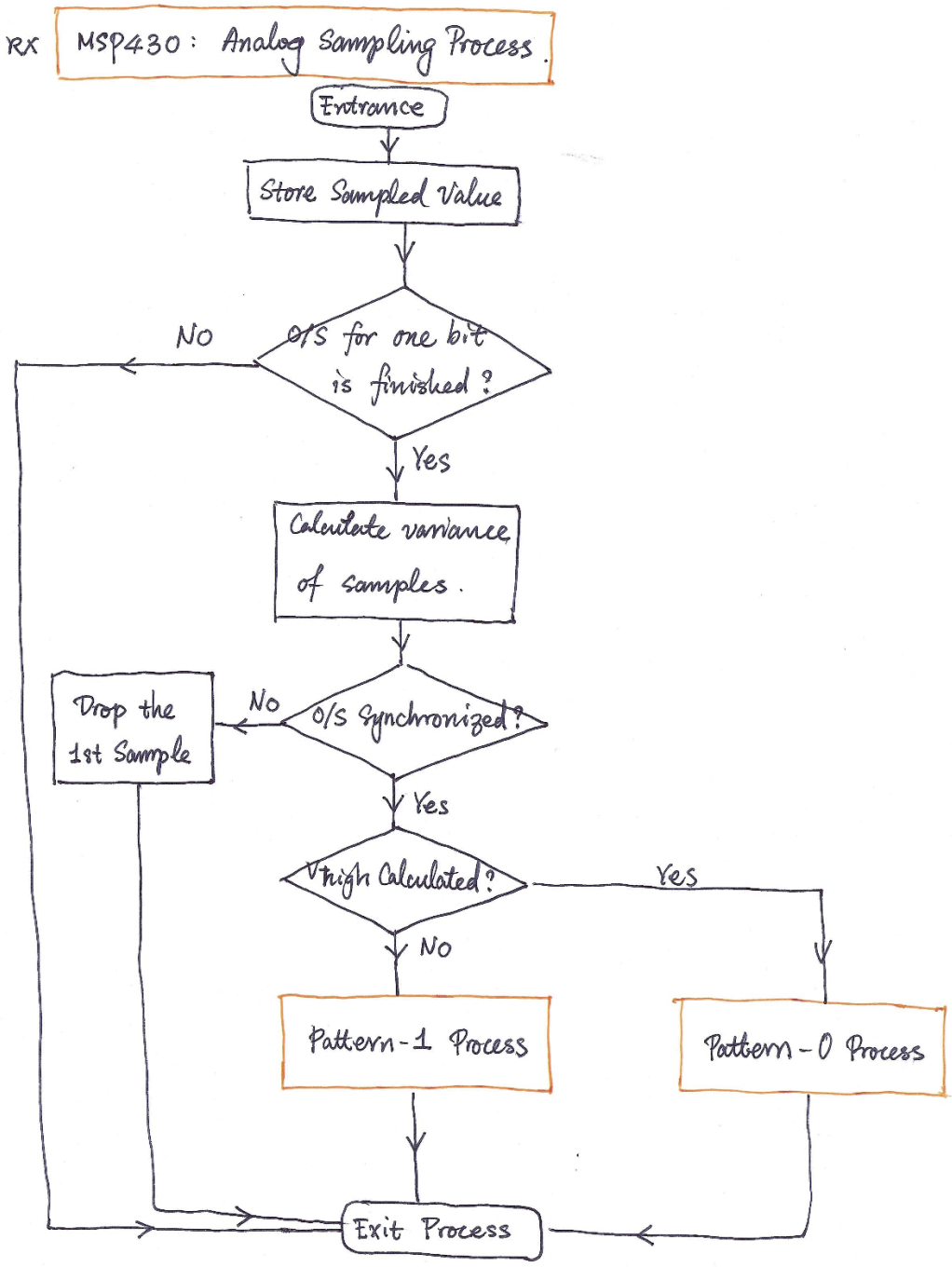


Figure 8: The analog sampling process.

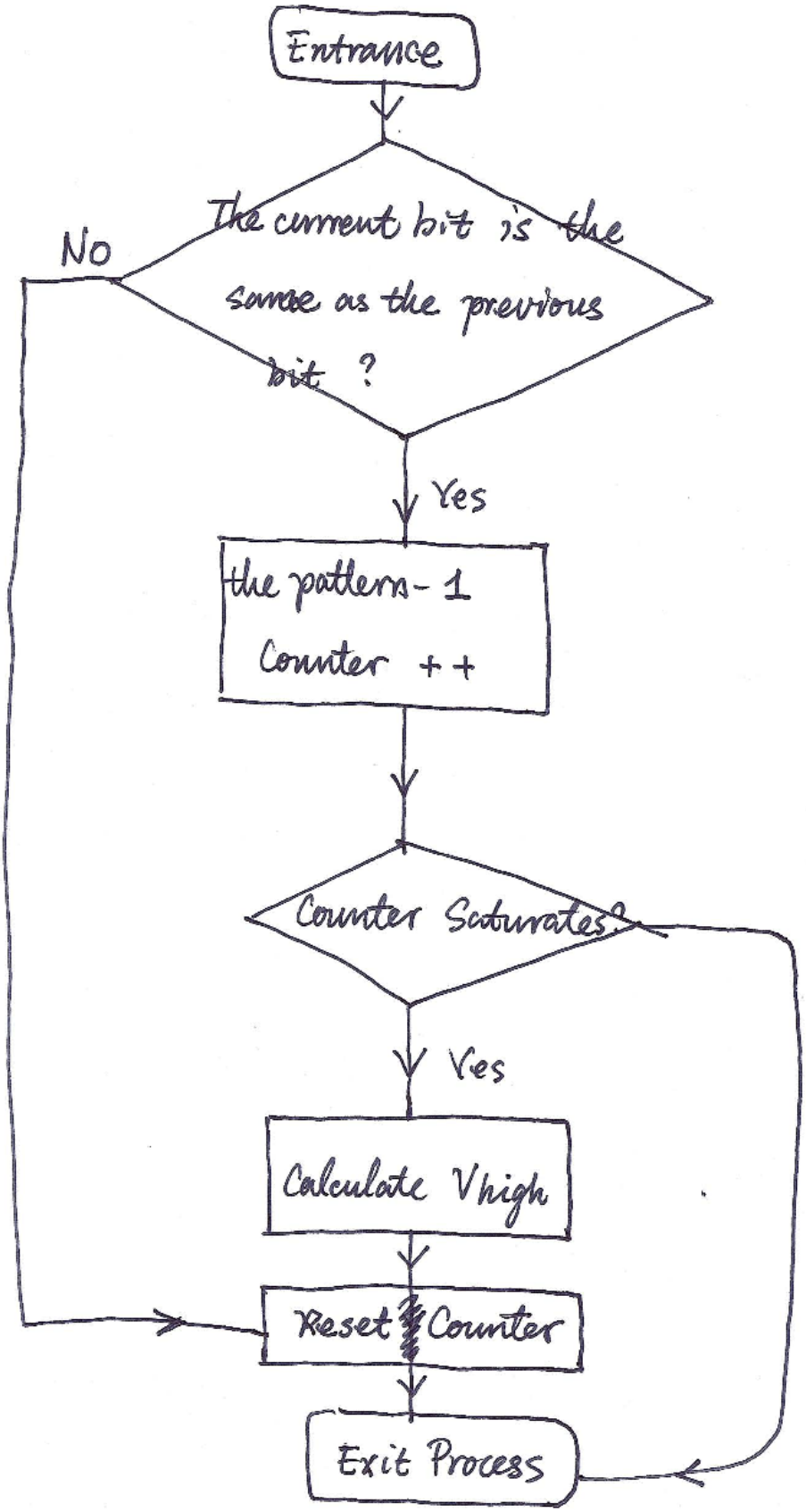


Figure 9: The pattern "1" sampling process.

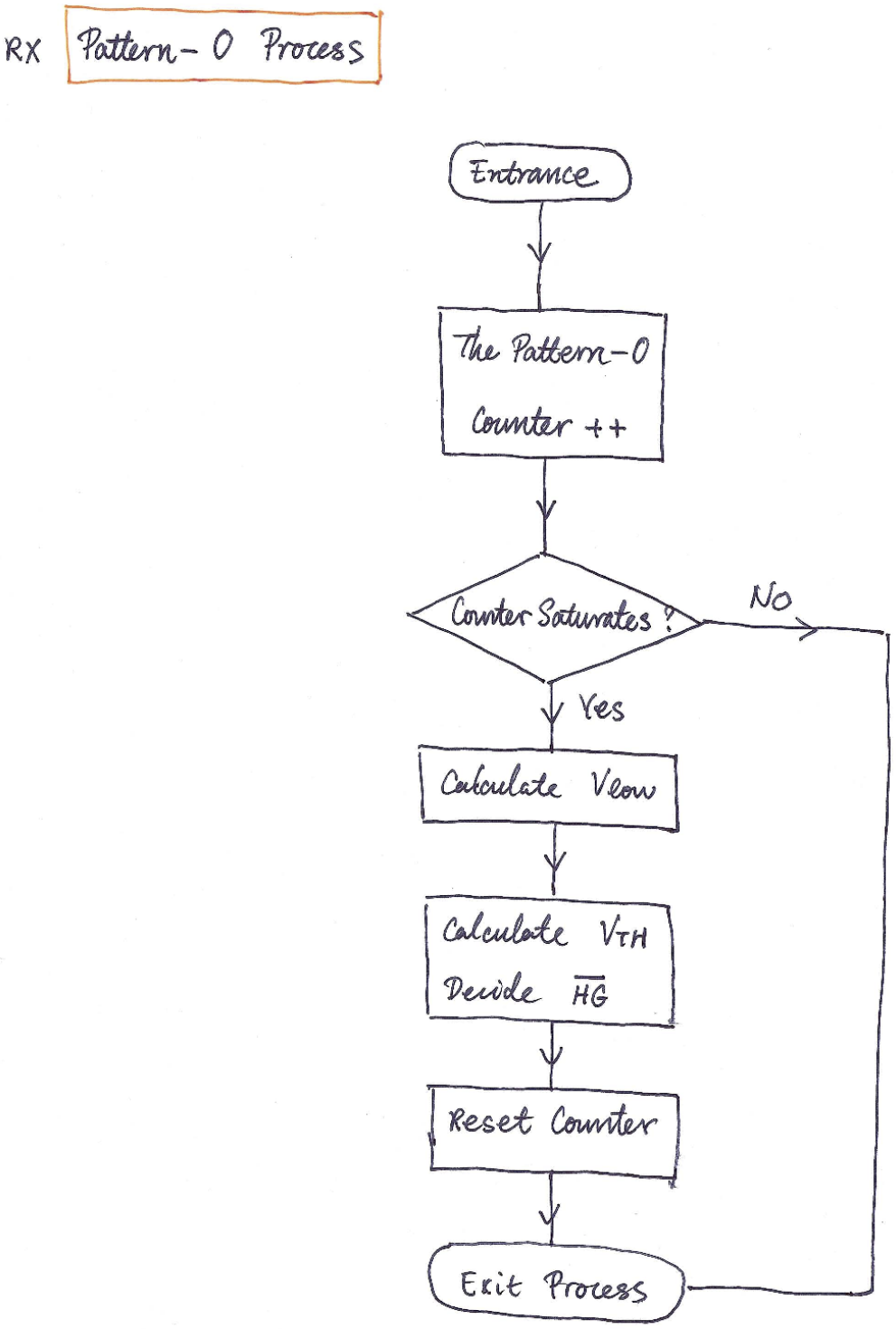


Figure 10: The pattern "0" sampling process.

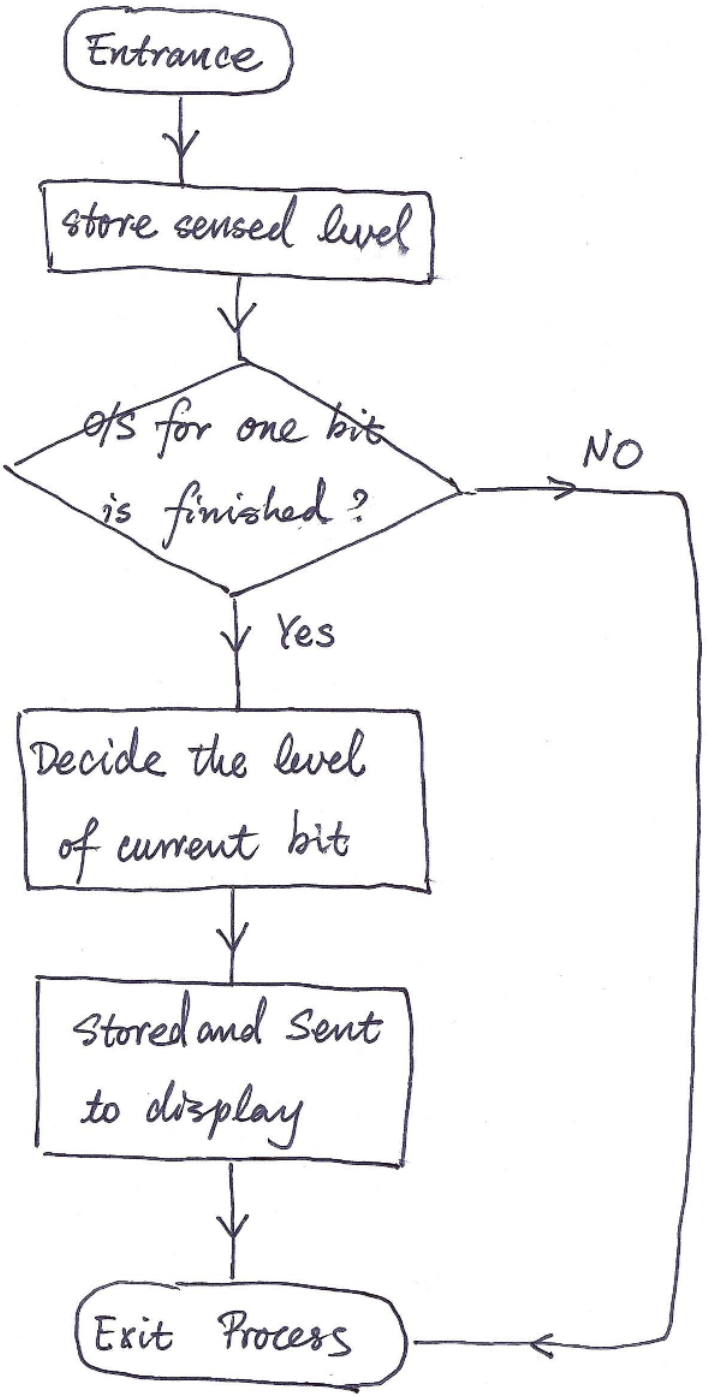


Figure 11: The digital sampling process.

6.1 Prelab/Lab Procedure

Week 1: Prelab: Carefully read this manual and examine the microcontroller pseudo-code. Write your own "C" code implementations of the functions outlined in the description. The transmitter code will be given to you for reference. If you do not understand any part of this manual, please consult with your lab GSI.

During your lab section, sketch a schematic for the various analog components, including the sensor, the I-V block, the buffer, the VGA, the ADC, and the DAC. Spend about 1 hour on this part of the project and once the GSI has approved your schematic, begin constructing your analog front-end.

Week 2: During the second week you should begin testing your analog front-end. First test it as a stand alone block and make sure it functions as you expect. You can replace the light sensor with a known input voltage and test the functionality of the VGA and ADC. Using the microcontroller and a volt meter, test the functionality of your DAC. Also begin testing your receiver code section by section. Can you successfully read in an analog signal, used in the analog signal processing section? Can you successfully read in a digital signal? Test the functionality of the header calculation routine by sending in known waveforms such as ramp signals. By the end of the second week you should be confident that all the sub-components (software and hardware) are functional. Also, the head GSI's 'C' code receiver routine will be available for comparison or last minute debugging.

Week 3: Combine all functional blocks to test your system. You should first test your system using a wired link just so that you can ensure everything is functional when a known input voltage is fed into the microcontroller. Next test the system with the wireless link. As you do these tests, you will find many problems with your design which may force you to go back and test the various sub-components. This is by far the most challenging aspect of the design since unintended interactions can cause problems. If all the parts were working correctly on their own and the interface between the blocks is well defined, then everything should be working. If you have doubts about your code, you may test your system with the GSIs software code. To complete the project, you should successfully detect an unknown pattern sent by the GSI using a purely wireless link (even the grounds should be separate).

6.2 Bonus

There are several ways in which you can improve the project. If you finish the project early and demonstrate successful data reception to the GSI, you are free to explore the design in any way you wish. Here are a few suggestions.

1. You can improve the range of the system by finding the average signal level during each bit, rather than by just sampling a single point in the waveform. Design an analog circuit to perform this operation (hint: try an analog integrator but dont forget to "dump the data at the end of the period). You can also do the same thing digitally by doing all the signal processing in the digital domain (use the on-board ADC). Confirm that the range for the system improves.

2. You may also try to add the capability to synchronize the clocks of the receiver and transmitter. One way to do this is to oversample the receiver and latch the input data

with the optimal edge of the clock. The program should try to determine if the edge was early or late and then generate a new clock at a slower rate (the actual data rate) which is optimal (transition occurs at the midpoint of the data period).

3. Try adding more bits to the VGA to increase the dynamic range of the system. Gain control is needed to avoid saturating the system when the transmitter is held too close to the receiver. Large gain is only needed when the receiver is far away and the received signal strength is so low that it cannot exercise the full dynamic range of the system (the transition from 1 to 0 is occurring in a voltage range less than the LSB of the system).

4. By the end of the class you will be familiar with many different kinds of filters. Try using a filter at the receiver to improve the signal-to-noise ratio. What is the optimum filter bandwidth?

5. Add a parity bit to the transmitter code and generate a longer packet. Redesign your receiver to detect if a packet error has occurred. Verify that the packet error rate (PER) goes up with distance.