Infrared Digital Scoreboard

CS272 Midterm Project – Fall 2017

Due: Monday, October 30, 2017 at 10:00am

1 Introduction

Please read this entire document. It is lengthy and contains many details – but most of these details are meant to make this project easier. Save yourself some headache and read this first.

This is the midterm project for CS272. This project will be a group project, in groups of two or three, assigned at random by me. To keep group members from slacking-off, **there will be peer grading** submitted by each member of the group evaluating the other group member(s). You will be given some time in class to work on this project, but you are also expected to coordinate with your group and work on the project outside of the class time. The lab is available with swipe card access using your student ID after hours.

The purpose of the project is to use some of the circuit techniques we've worked on so far this semester, practice writing microcontroller programs, and integrating circuits with the microcontroller to make a fully engineered system. The focus of this particular project is on the timing needed to communicate a value with another microcontroller, and to use a simple digital display to view the communicated value.

In this project you will develop a system that will display two digits of a scoreboard. These two digits represent the score for one of the teams in a game (for a variety of sports). The scoreboard will receive a score update from a second microcontroller via an infrared (IR) receiver BJT. The second microcontroller will be provided by the me. When your microcontroller receives a new score, it must interpret what value was sent, show a goal celebration pattern on the digital displays, and then update the displays to show the value that was received.

You are responsible for making sure all voltage, current, and power dissipation values in your circuit are within the rated specifications of the various circuit elements. You can reference your previous work in lectures and labs for examples of each circuit required for this project, and for calculations needed to determine circuit component values.

While it is not a requirement, it is highly advised to also split the microcontroller program into classes. Possible class types could be used to represent one of the scoreboard digits (and instantiated twice, one for each digit). Another possible class could encapsulate the logic for handling the IR receiver.

The due date for this project is **Monday, October 30th, 2017 at 10:00am**. Full submission of the project consists of five pieces: 1) a demonstration of your system to me, 2) submitting the circuit diagrams **and** the calculations that you used to prove that circuit elements are within their rated capacities, 3) submitting the Teensy source code, including any C++ classes, 4) leaving your breadboard assembled after so that I can ensure that each circuit is connected as intended, and 5) submitting a written evaluation of your groupmate(s) on D2L. Late projects cannot be accepted. It is in your best interest to start this project as soon as possible. All of the material that you need to know to complete this project can be found in lecture topics 1 through 12 (up through BJTs).

2 Project Specification

This project requires a microcontroller program and circuit that is able to communicate with another microcontroller. Communications often involve accurately timing the sending and receiving of a series of high and low voltages on wires that varies over time. The number of signals, and the pattern of high/low pulses is collectively referred to as the signaling *protocol*. In this project, we will use an IR LED/BJT pair to communicate the high/low pulses and will use a simplified RS-232 (serial) signal pattern for the protocol.

Although this project involves communicating between two microcontrollers, you are only responsible for the microcontroller that *receives*. I will supply a microcontroller that will send. Furthermore, instead of communicating

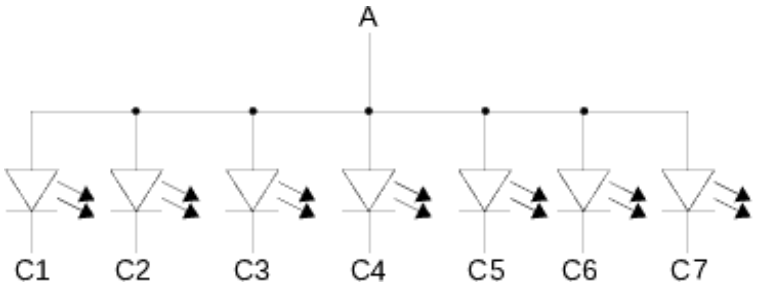

Figure 1: Internal circuit of a 7 segment display.

over wires, we will communicate wirelessly via infrared where high and low voltages will be represented by presence and absence of IR. Since my microcontroller will send the IR, my microcontroller will have an IR LED. Your microcontroller will receive, and thus your microcontroller will use an IR BJT. The following sections describe the circuits needed for your microcontroller, as well as the protocol that you should use to receive score updates.

2.1 IR Receiver

My microcontroller will emit infrared pulses in a specific pattern (outlined below). Your microcontroller must detect these pulses, and thus you will need to use an IR detector/receiver circuit. The IR receiver acts as a BJT, where the voltage of the base terminal is controlled by the amount of infrared light is incident on the lens of device. With more infrared light, there is a higher base voltage.

You should use an LTR-301 IR receiver BJT for this project, arranged in a *switching* configuration (as opposed to an amplifier). Thus, the output of the BJT should either be a high or low voltage. The datasheet for the LTR-301 can be found at `http://optoelectronics.liteon.com/upload/download/DS-50-93-0013/S_110_LTR-301.pdf`. The maximum collector current I_C is 0.0002A (0.2mA). You must determine what resistors that you will use along with the LTR-301 IR BJT. You must submit the calculations that you used to determine the resistance(s), as well as proof that your selection limits the collector current to an acceptable maximum and that the resistor is within its maximum power dissipation.

The circuit arrangement for the IR BJT can be similar to the switching examples on slide 20 of topic12.pdf. The V_{out} can be connected directly to a digital input pin of your Teensy, and the on/off status determined with the `digitalRead()` function. You can supply the IR BJT with the 3.3V supply pin of the Teensy. Be warned that ambient light (including florescent lights) may also emit infrared. If you suspect that the classroom or lab has stray IR that is making it difficult to turn the IR BJT off, you can cover your project or turn off the lights.

For testing, you can assume that there will be no obstructions between my IR emitter, and your IR detector. I expect that your project will function if the IR emitter and detector are a few inches apart.

Your project source code should implement a C++ class that handles receiving the IR pattern, and converts that pattern to two integers, one for each digit of the digital display. This will be further discussed below in the "Signal Protocol" section.

2.2 Digital Display

For this project, you will need two *seven segment displays*, each of which can display a single numeric digit from zero through nine. Seven segment displays are made of seven red LEDs. The anode terminals are all connected to a common pin of the display, and the cathode terminals are the remaining pins of the display. The schematic for the seven segment display is shown in Figure 1.

Each LED of the 7 segment display has a forward (barrier) voltage of 1.7V. You must determine how to connect these LEDs to the Teensy such that your Teensy can turn on/off each LED individually. Remember that the anode voltage must be greater than the cathode voltage by at least 1.7V to turn the LED on. You must use the `digitalWrite()` function to turn the LEDs off and on. You must also have a current limiting resistor – the value of which you must prove to properly limit current to 16mA, and does not dissipate more than the 0.25W rating of the resistors used in lab.

It is your choice which Teensy pins to use for which segments. It is also up to you to determine how to display each decimal digit from one to nine. The mapping in Figure 2 shows which pins of the 7 segment display are the cathode terminal of each LED. You do not need to light the decimal point LED of either display for this project. Assume that

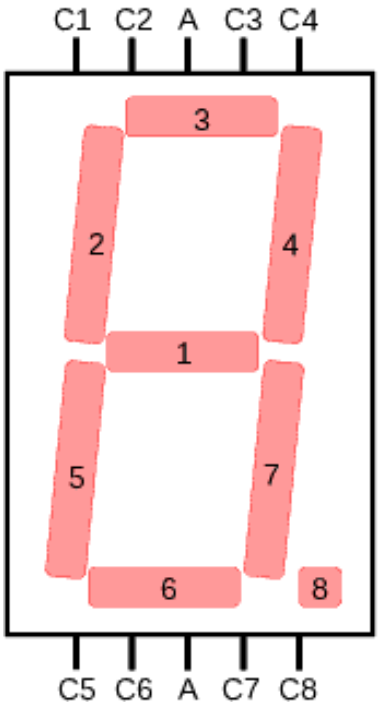

Figure 2: Mapping of LED segments to the pins of the 7 segment display.

scores are 0 at start-up. More information about how the scores should be displayed is discussed below in the section titled "Score Updates".

2.3 Signal Protocol

The pattern of high and low pulses of IR, used to communicate values, will loosely follow the RS-232 signaling protocol. This protocol is used to send a single, 8-bit value at a time.

Before a new value is transmitted, the IR light will be on, and thus, your IR BJT will be on as well. The start of a new value is indicated by turning the IR light off, and thus your IR BJT will turn off.

Once this on to off transition occurs, you must measure 10ms (10 milliseconds). After 10ms, you can read the status of the IR to determine the value of the most-significant bit of the 8-bit number (the 7th bit position). For this protocol, the presence of IR indicates a 1, and absence a 0. After another 10ms, you can read the status of the IR again to determine the next bit of the 8-bit number. Every 10ms, you can measure the IR to get the next lower bit position until you reach the least significant bit (the 0th bit position). Thus, it takes about 80 milliseconds to transmit a complete value.

After the last bit has been transmitted, the IR will be turned back on until the next 8-bit number should be transmitted.

Figure 3 shows an example timing diagram of the complete transmission of a single value. Timing diagrams show the voltage on the y-axis (simply HIGH and LOW for digital binary values), and time on the x-axis. As can be seen, the IR is on initially. Eventually, the IR is turned off – labeled "Start" in the diagram. After 10ms, it is safe for your program to read the bit value for the most-significant (7th) bit, in this case, it happens to be 0. Ten milliseconds later, it is safe for your program to read the 6th bit, another 0 in this example. After 80ms total, your program should have read all of the 8 values. In this case, your program should have read 00011010_2, which is equivalent to 26_{10}. Once this sequence is complete, the IR is turned on in preparation for the next value transmission (not shown).

Since this protocol sends 8-bit values, the possible range for each transmission is 00000000_2 to 11111111_2, which is 0_{10} to 255_{10}. However, since we only have two decimal digit displays, we will only be able to represent 0_{10} through 99_{10} in this project. This range corresponds to the binary values 00000000_{10} through 01100100_2.

This protocol will be used to send updated scores to your microcontroller. Each transmission of a value using this protocol will signal to your microcontroller to update both 7 segment displays.

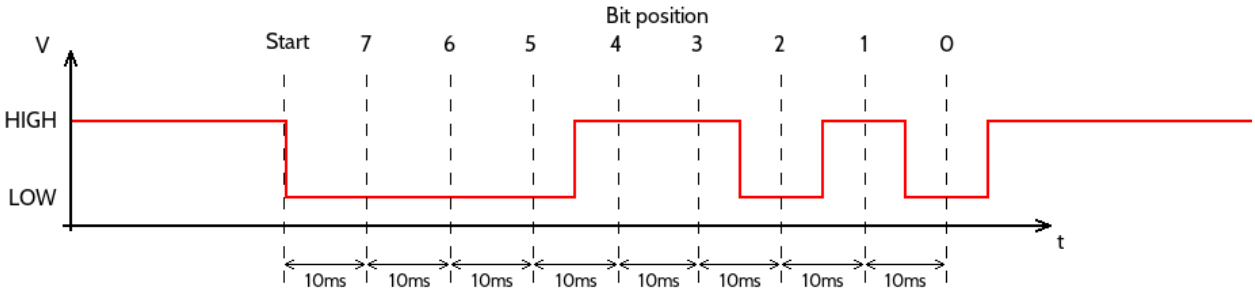

Figure 3: Example timing diagram showing the pattern used to transmit the value 26_{10} (0x1a).

Figure 4: Allowable digits for numerical scores.

For this project, a new value will be sent to your microcontroller at least 5 seconds after the previous value was sent. This 5 seconds is a lower-bound (you do not need to handle new scores being sent sooner than every 5 seconds). There is no maximum time between new scores.

2.4 Score Updates

As mentioned previously, the score displayed should initially be zero until the first score is received. If the score is less than 10, then the leading digit should be completely turned off. Digits should appear as shown in Figure 4.

Rather than immediately updating the scoreboard when a new score is received, your 7 segment display should display a short "celebration" pattern for both digits. When a new score is received, this pattern should be displayed step-by-step simultaneously by both digits, and once the pattern is finished the new score should be displayed. Figure 5 shows the pattern of segments for the celebration pattern. Each step of the pattern should be displayed for 100ms before moving to the next step of the pattern. This pattern will look like an LED victory lap when played.

After the 6-step pattern is played (600ms total), then the new score should be permanently displayed until the next score update is received.

3 Software Architecture

Good software engineering practice means that you should think about splitting this program into classes. While splitting your program into classes is not a requirement for this project, it is highly encouraged.

For example, you may want to move the logic that extracts a value from the IR BJT into its own class. This class could have a function that returns 0 as long as no new value has been received. When a new value has been received, the function then returns that value. The main `loop()` function can then just repeatedly call this function, do nothing when no new value has been received, and then update the displays when a new value is ready.

You could also split out the functionality needed for one of the 7 segment displays. This could be instantiated twice, one for each of the displays. This class could have a function that is called when a new value is received. That function

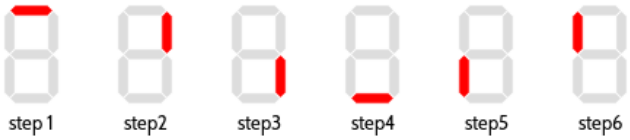

Figure 5: Segment pattern for the celebration of a new score.

could implement a state machine that progresses the LEDs through their celebration pattern, and once completed is updated to the correct decimal digit. Be careful using `delay()` statements inside of this function, however, since the function will not return until the delay is complete. I would suggest leaving the delay outside of this class, and only using this class to update the steps of the pattern. This function could then be called for both displays, then a delay used in the `loop()` function.

These are only possible ideas for the overall design of the microcontroller program. You are free to implement your program any way that you see fit.

4 Tester Microcontroller

To help you test your program, a microcontroller will be provided that will periodically send new score updates via an IR LED. This microcontroller has an LCD screen that will display the most recent score that was sent and is updated every 8 seconds with a new score (the score will increase by a repeatable pseudo random amount).

This microcontroller is powered with a 9V battery, and does not need to be connected to a PC. There is an on/off toggle switch to turn the power on and off.

The tester microcontroller also has a push button that, when fully pressed and released, will pause the microcontroller so that it does not send out a new score until un-paused. You can just press and release the button again to un-pause.

There will only be a single tester microcontroller! You must share the tester with other groups. Please be courteous. Also, plan ahead... you may not have immediate access to the tester if you put this project off until the last day!

5 Grading

As mentioned above, grading is based on five components. First, you must demonstrate your project before the deadline to me. The scoreboard must be updated to the correct values, also displaying the celebration pattern. The pseudo random score increments will be different than the pattern used by the project tester.

Aside from the demonstration, there are several items that must be submitted. First, the circuit diagrams along with calculations that you used to determine resistor values, the current through the various components, and power dissipation of the resistors should be submitted. You can either submit on paper, or on D2L.

You should submit all of your source code (the .ino, and both sets of .h and .cpp files) to D2L. Please zip these source code files into a single file, and then submit the zip.

You should leave your assembled breadboard(s) with me for grading. While this project is not a beauty contest for pretty wiring, it is expected that your breadboard(s) can be moved without being disassembled, and without wires coming unconnected, etc. You may lose points if your breadboard is so untidy that it does not work after being picked up and moved. It is in your best interest to use plain wire (not the jumpers) that has been trimmed to proper lengths. I will dismantle your breadboards once grading is completed.

Finally, you should submit an evaluation of your groupmate(s) on D2L. This should be roughly one paragraph that describes your role in the project (i.e. what parts you worked on) as well as your partner's role in the project (the parts he/she worked on). You should also describe how well both you and your partner performed at these tasks, how active both of you were in attending group meetings, coding sessions, circuit debugging, etc. A group member that is not fully participating and fully contributing may lose points relative to the other partner.

The D2L dropbox for this assignment is named "Midterm Project".