Smart Light Switch

CS 272 Midterm Project – Fall 2019

Due: December 11, 2019 at 12:05pm

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 final project for CS 272. You will be given *some* time in class to work on this project, but you are expected to work on the project mostly outside of the class time.

The purpose of the project is to use many of the circuit techniques we've worked on this semester, practice writing object-oriented microcontroller programs, and integrating circuits with the microcontroller to make a fully engineered system.

In this project you will develop a smart light switch that will detect when at least one person enters a room and will automatically turn on a lamp. After the last person leaves the room, the lamp will automatically turn off. The system should also have a push-button to manually turn the lamp on or off, a timer to turn the lamp off after an adjustable amount of time, as well as a lamp brightness adjustment determined by the position of a potentiometer.

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.

The due date for this project is **Wednesday, December 11th, 2019 at 12:05pm**. Submission of the project consists of four pieces: 1) your project must be demonstrated to me before the deadline, 2) the circuit schematics and voltage, current and power calculations must be submitted either on paper or on Canvas, 3) the Teensy source code must be submitted on Canvas, and 4) you should leave the breadboard with your circuits with me – you will get it back at the final exam. 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 has already been given in lectures. Keep in mind that it is the end of the semester, and I may not be available the entire day for demos.

2 Project Specification

I'm sick and tired of the wonky, erratic behavior of the automatic light switch in the Wing 246 lab room. I promise that you – the well-equipped, motivated, and creative CS 272 students – can do much better. UWL CS 272 students to the rescue!

You should engineer a smart light switch system that has the following three methods of operation:

1. A set of infrared (IR) LED/BJTs mounted on a doorway that can be used to detect when a person enters a room and then turns on the lamp automatically. As more people enter the room, the system keeps track of the number of people in the room. As people leave the room, the system will automatically turn off the lamp once the last person leaves.

2. You cannot always guarantee that users will exit the room the same way in which they enter. Perhaps they leapt out of a window. Maybe they exited the room through a different door. Probably they more likely leapt out the window. We don't want our lamp to stay on forever in this case. Our system should keep track of the time since the last (most recent) person entered the room. If the lamp is on long enough that an upper-bound time is reached, the lamp should turn off automatically. The upper-bound can be varied by the turning the knob of a potentiometer.

3. The system still allows for a curmudgeon user who doesn't want a fancy new-fangled light switch. The system caters to this type of user by providing a push-button that can be used to manually turn the lamp on or off like a normal light switch.

In addition to these modes of operation, the system should allow for a lamp brightness adjustment. There should be a second potentiometer that can be used to turn up or down the lamp brightness any time the lamp is lit.

Your system should be based on the Teensy 3.2. The project can be split into these sub-systems, but can all be controlled using a single Teensy. Each sub-system is described in more detail in the following sections.

2.1 Lamp

The lamp for this project is *not* going to be an LED, but instead will be an incandescent lamp. The lamp can be safely operated at 9V and draws about 0.25A of current at full brightness. Of course, the pins of the Teensy are not capable of supplying either the required voltage or the required current. Thus, you must use an external 9V battery. The lamp can then be turned on and off using a MOSFET or a BJT (your choice), arranged similar to the motor in lab 4. The lamp does not have appreciable inductance, so the flyback diode can be safely omitted.

If you choose to use a BJT, you will want to use the 2N4401 BJT from the cabinet (not the BJT in your lab kit) because it allows a higher I_C current, needed by the lamp. Its datasheet can be found at `https://www.mccsemi.com/pdf/Products/2N4401(TO-92).pdf`. If you use your MOSFET, you can use the one in your lab kit, its datasheet can be found at `https://www.onsemi.com/pub/Collateral/FQP30N06L-D.pdf`.

If you're curious and want to see the datasheet for the lamp, it is part number CM2181 and can be found at `http://static.vcclite.com/pdf/T-11_2WedgeBase-T-13_4WireTerminal.pdf`. However, the lamp datasheet is not very helpful. Note that the 6.3V rating is a *minimum* voltage needed to turn the bulb fully on, but they are safe operate at 9V. You can find the lamps in the right side door in the cabinet. Please be careful with the lamps, they can get hot when turned on fully for long periods of time. Also, there are only enough lamps for everyone to use one.

2.2 IR Emitter/Detectors

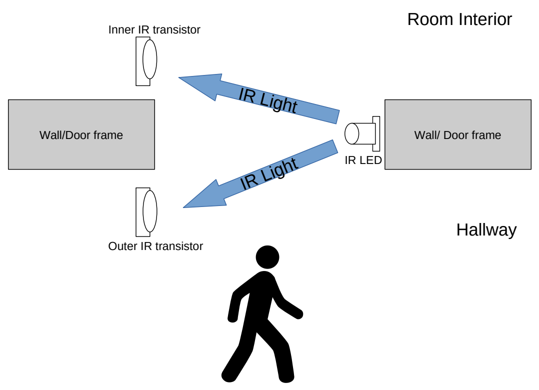
To detect when people enter and leave a room, your project should include an IR LED, and two IR transistors. Figure 1 is a diagram that shows how the IR LED (emitter) and transistors (receivers) can be positioned. The diagram shows the entrance to the room as if you were looking down on the room from the roof. The IR LED is placed on one side of the door opening, and the two IR transistors are mounted on the other side of the door opening, one on the inside of the door, and one on the outside.

Normally, when no one is in the doorway (Figure 1a), the IR transistors both have the IR light from the IR LED shining on them. When a person starts to walk into the room, the outer IR transistor will have its IR light from the LED interrupted (Figure 1b). As the person continues to move into the room, the second IR transistor will have its IR light from the LED interrupted (Figure 1c).

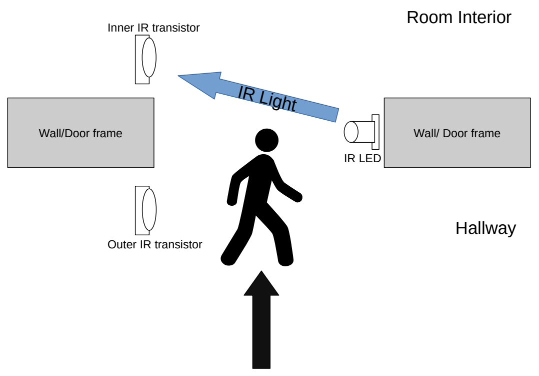
Once this sequence of events – the outer transistor has its IR interrupted, followed by the inner transistor interrupted – has occurred, your system can turn on the lamp and keep track that there is one person in the room (Figure 1d). If the same sequence occurs, then your system will keep the lamp on, but keep track that there are now two people in the room. This can repeat for as many people enter the room. During testing, it is safe to assume that no more than 100 people will be in the room at any given time.

If the IR transistors are interrupted in the opposite order – that is, the inner transistor is has its light interrupted, followed by the outer transistor having its light interrupted – then the system will determine that a person left the room. If there were multiple people in the room, then the light should stay on. This can repeat as people leave the room. Once the number of people that enter the room reaches zero, then the system should turn off the lamp.

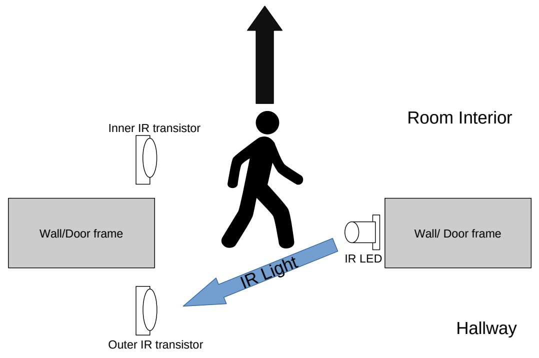
The IR LEDs that should be used in this project are generic IR LEDs (I don't have a part number). Their barrier voltage is about 1.2V, and their maximum forward current is about 0.05A. The IR LED can be supplied by the 3.3V Teensy supply pin (next to pin 23). You must limit the current through a resistor. You can find the IR LEDs in the right side door of the cabinet. The IR transistor is part number LTR-301, and its datasheet can be found at `http://optoelectronics.liteon.com/upload/download/DS-50-93-0013/LTR-301.pdf`. These are the same IR transistors that were used for the midterm project. It has a maximum collector (and hence, also emitter) current I_C of 0.0002A (0.2mA). It is not necessary to power the IR LED or transistors with the 9V battery, the 3.3V



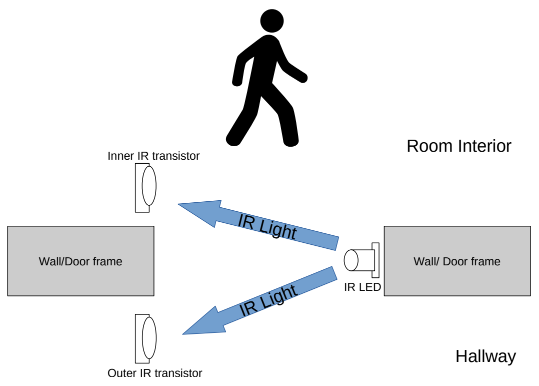
(a) Initially empty room.



(b) First IR transistor has IR light interrupted.



(c) Second IR transistor has IR light interrupted.



(d) Person has fully entered room.

Figure 1: Sequence of actions showing one person entering a room.

supply from the Teensy is sufficient. You must show the calculations to show your selection of resistors, as well as showing that the power dissipated by the resistors is within the 0.25W rating of the lab resistors. You must also show your schematics.

To make testing easier, Figure 2 shows the orientation of the IR LED and transistors, with example distances indicated. The LED and transistors are closer together than you would expect for an actual doorway, but I don't want you to run into problems with the range between the LED and transistors, IR reflections, angles of incident light, etc. This makes it easier to get these LED/transistors to work.

Also when testing, I will interrupt the light between the LED and transistors using a solid object, for example a piece of cardboard. I have found that a flat-black piece of cardboard works best. Using this solid object to obscure the IR means that you don't have to worry about problems stemming from partially obscured light. The IR light will be fully cut-off between the LED and transistor. Additionally, the IR light will be interrupted for one transistor, then uninterrupted again before interrupting the light for the second transistor. This means that you don't have to worry about both transistors having their IR light obscured simultaneously.

There is no maximum limit on 1) the time from interrupting the light to it being uninterrupted again, and 2) the time between interrupting the first IR transistor compared to the light being interrupted on the second IR transistor. Your program must be able to handle any amount of time between interruptions. I will, however, have a minimum time between these events – 500ms (half a second). This is not meant as a value for you to hard-code into your program... it means that your system does not need to be able to react to someone entering the room extremely fast. This restriction just means that you probably **do not** need to take any special timing into consideration, and also means that your Teensy program does not need to be very high performance. In this project, I want your system to function correctly, performance is a secondary concern.

Finally, when testing, we will be sure that there isn't a lot of sunlight in the room. Sunlight has a significant amount of infrared, and thus could cause your IR transistor to always appear to have IR light incident on the lens even with a solid material placed between the LED and transistor. The light from the florescent lights in the lab/classroom will not

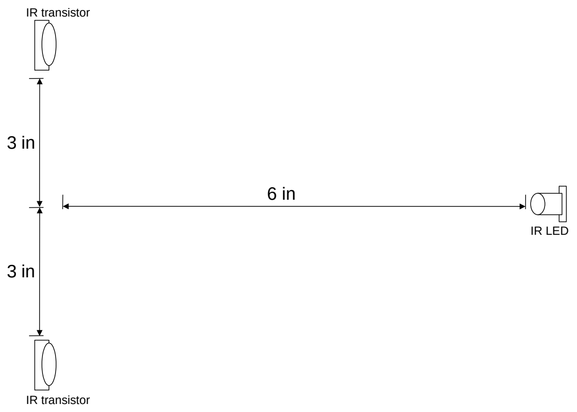


Figure 2: Spacing between the IR LED and transistors to be used during testing.

likely cause any problems.

2.3 Timer

When a person enters the room according to the steps in the previous section, your program should keep track of the time. You can use the `Metro` library to help keep track of time – more on this later. As long as at least one person is in the room, the time should be continuously checked, and if time exceeds a limit, then the lamp should automatically turn off and the number of people assumed to be in the room should be reset back to zero. In this case, we assume that someone snuck out of the room (through another door, the person ducked under the IR LED/transistors, etc).

The time limit should start over as new people enter the room. Thus, the time until the lamp automatically turns off is based on when the most recent person entered the room.

The time limit should be determined by the angle on a potentiometer. The potentiometer can be part of a voltage divider circuit, whose output bias voltage is fed to an ADC pin of the Teensy. The Teensy can read the voltage using the `analogRead()` function, discussed in lecture topic 16, slide 20.

The voltage divider can span any voltage range that you wish, but should not go above the 3.3V supply voltage of the Teensy. The time limit should range between a minimum of 1 minute up to a maximum of 10 minutes. This is short for an actual system, but I don't want to have to wait all day during testing to make sure the timer limit is reached.

During testing, I will not simulate the possible scenario that the timer turned the lamp off before everyone has left the room.

Again, you must show your calculations for any resistors used in this circuit, and you must show that resistors are within the 0.25W maximum dissipation. And don't forget schematics. Recall there was a nifty trick with potentiometers such that you do not need any other resistors other than the potentiometer in order to have a voltage divider.

2.4 Dimmer

Your system should include a second potentiometer, again used in a voltage divider circuit whose output is read into the Teensy sketch. This second potentiometer should be used to dim the brightness of the lamp. If you use the same circuit for this second potentiometer, then you do not need to duplicate any calculations. You should, however, still show where your circuit will connect to the Teensy in a schematic.

Dimming the lamp is similar to slowing a motor. See lecture topic 16 slide 22-25. Also, watch out which pin you use to turn on/off the lamp – since only certain pins of the Teensy support PWM.

Your lamp must dim from full brightness at one end of the potentiometer to a minimum brightness at the other end of the potentiometer. You will need to experiment with brightness to see what minimum PWM value to use that can still be seen. The lamp should **not** completely turn off when the potentiometer is at its lowest setting!

You can find the potentiometers in the right-hand cabinet door, but again be careful because there are only enough for everyone to use two.

2.5 Push-Button

Finally, the system should allow for normal light-switch operation using a push-button. When the button is pressed and released, the lamp should toggle (i.e. turn on if it was previously off, or off if it was previously on). You can use the normal push-buttons that we've used in labs. Don't forget to include a pull-up or pull-down resistor to make sure that the Teensy input pin is not floating.

You must properly handle debounce. You can use your own hand-written debounce code, similar to the lab. Alternatively, for this project, you may use the built-in debounce library (called the "Bounce" library) that is available through the Arduino libraries. If you would like to use the built-in debounce, you can find more information at `https://www.pjrc.com/teensy/td_libs_Bounce.html`.

If the push-button is pressed, then the IR LED/transistor and timer inputs can be ignored. So if anyone is in the room when the push-button is pressed, then the lamp should still turn off. You do not need to worry about people entering or leaving the room if the push-button is used at any point – this scenario will not be done during testing. Once the push-button is used to turn the lamp off or on, then only the push-button needs to be checked for the remainder of the test.

When the push-button is used, the lamp should still have a brightness as indicated by the dimmer potentiometer.

3 Teensy Sketch

For this project, it is **required** that you have at least one C++ class. I suggest making C++ classes for each subsystem for this project. The constructors can take pin numbers as arguments, and the constructors can then set `pinMode` and do the `digitalRead` and `analogWrite` calls as needed.

Furthermore, it is **required** that you **do not** use the `delay()` function. You may use the `Metro` library for timing. Alternatively, you may use `millis()` to keep track of time. If you use the `Metro` library, I suggest having a single `Metro` timer that counts time at a low-ish interval, say 1 second. Then, every time the timer fires, you can call a C++ class function to handle the event. For example, a timer class could have a function that, when called every second, simply increments a variable that keeps track of how many seconds have passed since the most-recent person has entered the room. The class can also have a function that is called to reset the count each time a new person enters the room. Once the count has reached the upper-bound, as determined by the potentiometer (which may also be part of the same class), the function can return a value that indicates that the lamp should be turned off.

It is possible to have a global pointer variable that holds the memory address of a C++ class. However, one quirk of C++ is that any constructor called globally will execute before the `setup()` function. So you should not have any code that relies on `setup()` having been completed before the constructor – as this will not be the case.

4 Hints

You will probably want to work on one element of this project at a time. That way if you run out of time, you have as many elements successfully working as possible. This way you can also incrementally increase the complexity of the project, instead of trying to tackle every problem at once.

The easiest part of the project is likely to be the push-button activation of the lamp. You can start by implementing the push-button functionality so that it turns an LED off/on. Once that works, then you can take out the LED, and then add the lamp and BJT or MOSFET and have your Teensy turn the lamp off/on with the push-button. At this point, it might be best to add the dimmer potentiometer and add the PWM code. Then you can add the IR LED, and the IR transistors. The IR LED is easy, since it can simply stay on all the time. The IR transistors are more challenging, requiring that you keep track of the status of both transistors, and act according to interruptions in IR light. Once you have the IR LED/transistors working, you can work on the timer and timer potentiometer circuit. You can start by implementing a timer that has a hard-coded length of time. Then follow up with adding the potentiometer, reading the value with the ADC and adjusting the length of time based on the ADC value.

Stop by my office if you run into trouble. Do not wait until the last second to try to get help.

5 Grading

This project can easily be split into many subsystems that can be developed independent of the other subsystems. You will be graded on the correct functionality of each subsystem. That way, if you aren't able to complete the full project, you can still get partial credit for the subsystems that you were able to finish. This grading policy means that it is better to have correct operation of most of the system working – maybe one subsystem that doesn't function – than it is to have all subsystems on the board but none of them properly functioning.

Grading is based on the submission of four items. First, you must demonstrate your project to me before the deadline. This means that if there is a rush right before the deadline, you might not have time to demonstrate. So start early, and work often. Second, you must submit your calculations for current, voltage, and power for the circuit elements of your system. These can be submitted on paper, or on Canvas. Third, you must submit your Teensy source code on Canvas. The fourth item is to submit your project breadboard with all circuits intact.

Unlike the midterm project, a small percentage of the grade **will** depend on how tidy you make your breadboard (short wires, consistent wire colors, etc.). Also in the midterm project, each of the four grading items were equal weight – however, for this project, the demo and correct functionality of each of the subsystems will account for a higher proportion of the grade.