Original Abstract

I am planning on building a circuit that simulates a combination lock with flashing LEDs and an alarm system. The circuit will allow the user to set a four-input combination which will be stored in memory through the use of d-latches. Once the combination is set, the user can enter in a combination and if it matches the combination stored by the d-latches, a green LED will light up indicating to the user the combination was correct. If the combinations do not match, then a red LED will begin to flash and an alarm will go off, indicating that the combination is wrong and alerting anyone in the surrounding area that a potential theft may be in progress.

While possible to utilize MOSFETs to store the initial combination, it would be incredibly difficult and require many MOSFETs. As of now I plan on either using a 555-timer or using a flip-flop chip (SN74HC175) to store the initial combination. I hope to use MOSFETs to perform the rest of the logic that compares the code the user set to the combination that is currently being entered. However, the logic requires numerous XOR gates which can also be quite difficult to implement using MOSFETs so there is the possibility that I might have to use a XOR chip (SN74HC86). The rest of the logic requires mainly NOR gates which are quite simple to implement with MOSFETs.

For the red LED and speaker that will act as the alarm, I plan on utilizing a 555 timer in astable mode which will allow the red LED and alarm to repeatedly turn on and off, creating a sense of urgency. I may also use the LM386 audio amplifier to drive the speaker if necessary.

Operation

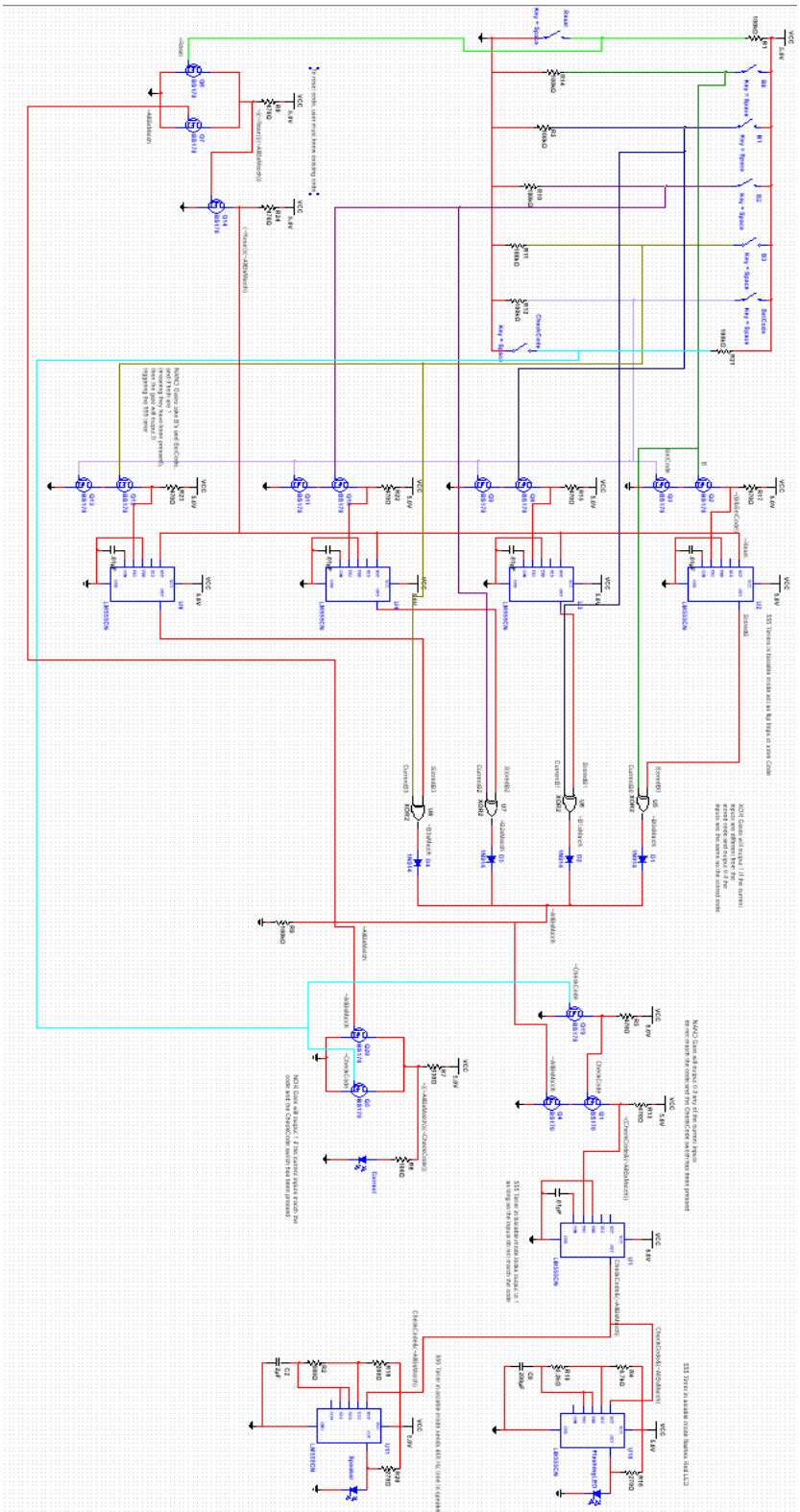

Figure 1 Full Circuit Schematic

User Input

The circuit utilizes seven switches that are all controlled by the user. The first switch shown on the schematic acts as a reset switch, and when it is closed it allows the current code to be cleared to 0000. However, the code is only cleared if the user knows the current code (explained in more detail later in the report). The next four switches (B0-B3) correspond to the bits of the code. Since there are four switches and each can be in the off state or on state, this allows there to be 16 possible codes. The SetCode switch allows the user to set a new code based on the current states of B0-B3. The last switch, CheckCode, allows the current states of B0-B3 to be compared against the stored code.

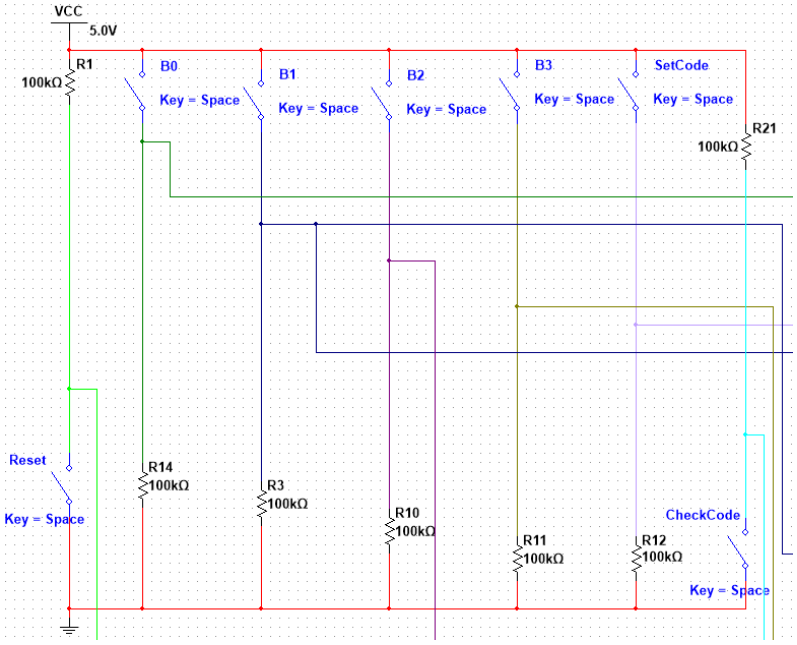

Figure 2 Switching Matrix

Storing the Code

The schematic to the right shows how one bit is stored with a 55-timer. The states of one bit of the code and SetCode are fed into a NMOS NAND gate and the output of the NAND gate is fed to the trigger pin of the 555-timer. If the bit and SetCode are equal to 1, the output of the NAND gate will be 0, and because the trigger pin is active low this will result in the

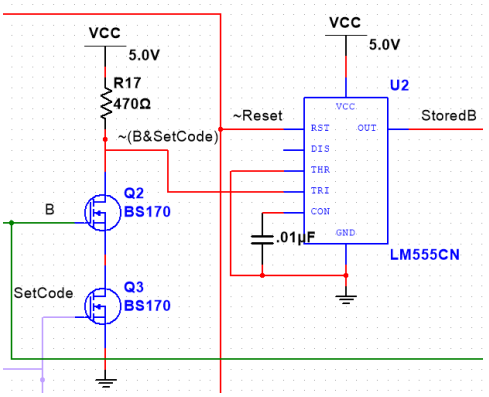

Figure 3 555 Timer in Bistable Mode Acts as Flip Flop

output of the 555-timer being 1. Otherwise, the output of the 555-timer will be 0.

Checking the Code

The schematic shows how one bit comparison is made. If the bits match, meaning they are both 0 or they are both 1, then the output of the XOR gate will be 0. Otherwise, the output of the XOR gate will be 1. The switching diode acts as an OR gate when multiple XOR gate outputs are tied together.

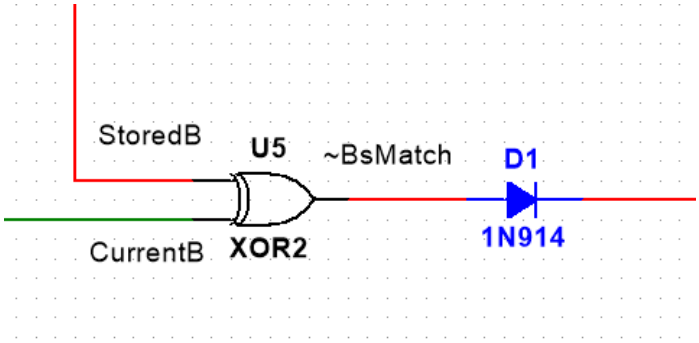

Figure 4 Comparing Bits Using XOR Gate

Code is Correct

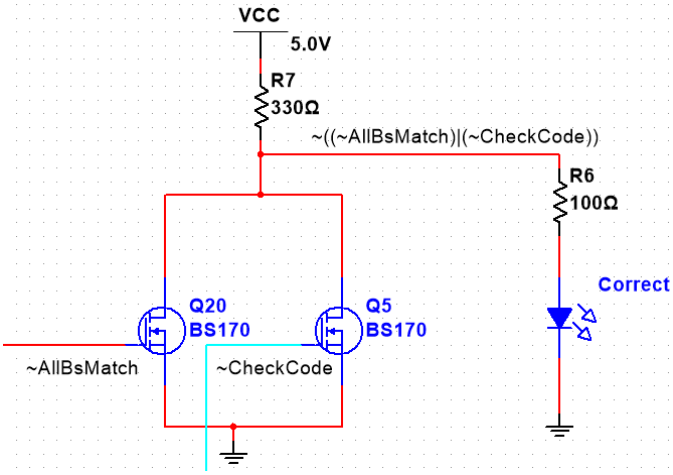

Figure 5 LED Will Turn On if Codes Match

If all the bits match the corresponding stored bits and the CheckCode switch is closed, then a green LED will turn on indicating to the user that the code they entered is correct. To perform this logic, an NMOS NOR gate is used with the output of the diode OR gate and the state of the CheckCode switch being used as inputs. If all the individual bits match the stored bits, then the diode OR gate will feed the NOR gate a 0. The CheckCode switch has an active low configuration, so if it is closed it will feed the NOR gate a 0. Both those 0s NORed together will output a 1, turning on the green LED.

Code is Incorrect

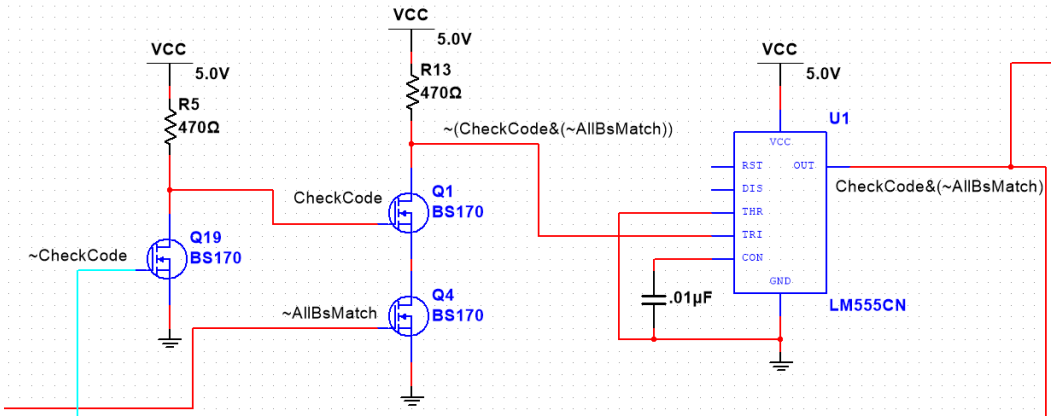

Figure 6 555 Timer in Bistable Mode Locks Alarm On if Codes Do Not Match

If all the bits do not match the corresponding stored bits and the CheckCode switch is closed, then a red LED will flash on and off and a speaker will turn on, together creating an alarm indicating to the user that the code they entered is incorrect. Once the alarm turns on, the 555-timer in bistable mode does not allow it to turn off until the code is correct. To perform this logic, an NMOS inverter and NMOS NAND gate are used. The inverse state of the CheckCode switch acts as one of the inputs to the NAND gate. The other input to the NAND gate is the output of the diode OR gate. The output of the NAND gate is then fed to the trigger pin of the 555-timer. For example, if the CheckCode switch is closed, the inverter will output a 1. If all the bits do not match the stored bits, the OR gate will output a 1. This in turn will result in the NAND gate outputting a 0 and triggering the 555-timer.

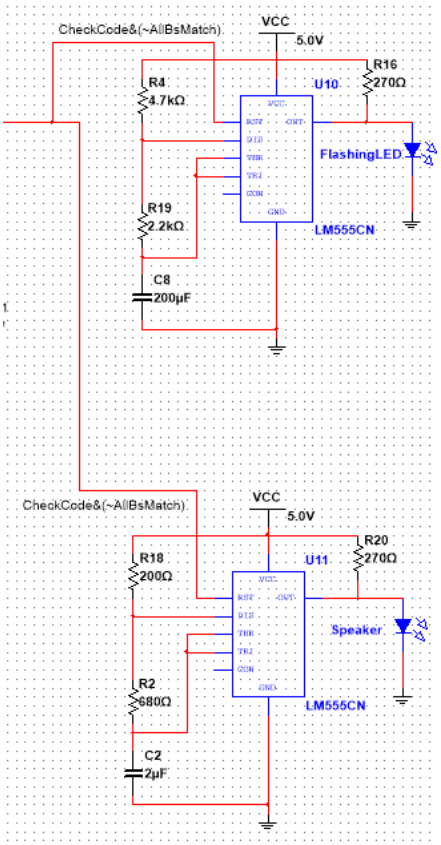

Figure 7 555 Timers in Astable Mode Drive LED and Speaker

Alarm System

To create the alarm, two 555-timers in astable mode are used. The 555-timer connected to the red LED outputs a signal with a frequency of about 0.8 Hz and a duty cycle of about 75%, meaning that the LED will be on for about 0.96 seconds and off for about 0.3 seconds each cycle. The 555-timer connected to the speaker outputs a signal with a frequency of about 440 Hz and a duty cycle of about 55%, meaning the speaker outputs a tone relatively close to a concert A.

Resetting the Code

To reset the code to 0000 the user must first know the current code. To achieve this logic, an NMOS NOR gate and NMOS inverter is used. The two inputs to the NOR gate are the state of the reset switch and the output of the diode OR gate. The output of the NOR gate is then inverted and is sent to the reset pins of the four 555-timers storing the code. For example, if the reset switch is closed, the active low configuration will result in a 0 and if the user is currently inputting the correct code, then the diode OR gate will output a 0. This will result in the NOR gate outputting a 1, which will then get inverted before being sent

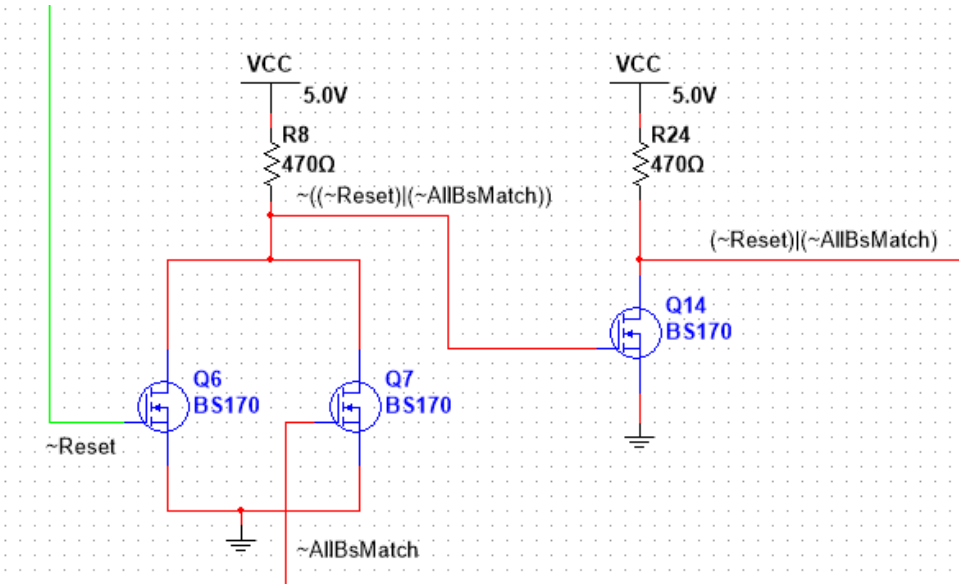

Figure 8 OR Gate Used to Reset Code

to the 555-timers. The reset pins on the 555-timers are active low, thus the code will be cleared to 0000.

Results

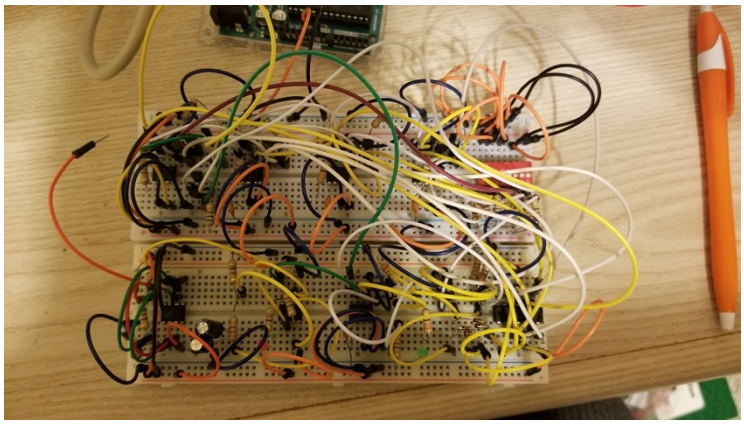

Figure 9 Full Breadboard Circuit

I was able to stick very closely to the plan I established in my abstract. The only feature I was not able to implement the exact way I wanted was the speaker. I initially wanted it to turn on and off at the same frequency as the red LED when the alarm was activated. I thought I would be able to achieve this by connecting the LED signal to the reset pin of the 555-timer connected to the speaker. However, this only seemed to distort the noise coming out of the speaker, so I decided to just let the speaker output a continuous tone while the alarm was active. Otherwise, I was able to add an extra feature I did not initially intend on having. For additional security, I added some logic that only allowed the user to reset the switch if they knew the current code, rather than allow anyone to simply close the reset switch to clear the code.

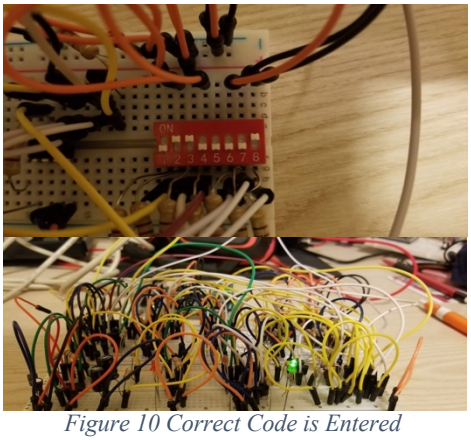

Figure 10 Correct Code is Entered

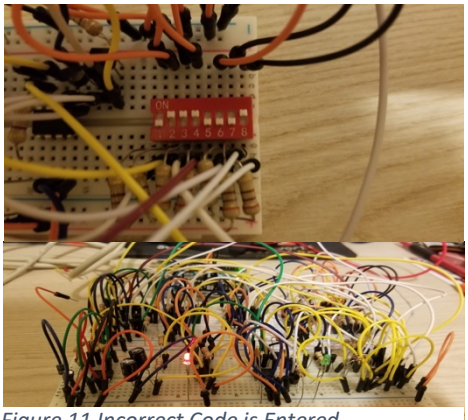

Figure 11 Incorrect Code is Entered

Applications

Physical security is an incredibly important field as people are always trying to figure out ways to steal valuables, information, etc. While a basic idea, if this circuit were integrated into a system with a physical locking mechanism, it would create a fairly effective way to ensure only those with the code are allowed to bypass the lock. The current circuit only allows for 16 unique combinations, which can all be tested very quickly. However, adding just four more switches (as well as four more NAND gates, 555-timers, and XOR gates) would allow for there to be 256 unique combinations making it much more difficult for potential thieves to brute force their way through the system.

Reliability

To calculate the probability the circuit will not fail during a one-year period, the Parts Count Ground Fixed model was used. The table below shows the generic failure rate ($\boldsymbol{\lambda}_{GF}$), quality factor ($\boldsymbol{\pi}_{Q}$), and part failure rate ($\boldsymbol{\lambda}_{P}$) of each component used.

Component	$\boldsymbol{\lambda}_{GF}$	$\boldsymbol{\pi}_{Q}$	$\boldsymbol{\lambda}_{p}$	Quantity	Quantity * $\boldsymbol{\lambda}_{p}$
555 Timer	0.024	10	0.24	7	1.68
XOR Chip	0.012	10	0.12	1	0.12
Switches	0.003	20	0.06	7	0.42
LEDs	0.0012	8	0.0096	2	0.0192
Capacitors, Ceramic	0.0074	10	0.074	5	0.37
Capacitors, Electrolytic	0.061	10	0.61	4	2.44
Resistors, Carbon Composition	0.0022	8	0.0176	25	0.44
Si FETs	0.099	8	0.792	15	11.88
Diodes, Switching	0.0075	8	0.06	4	0.24

Summing up the values in the **Quantity * $\boldsymbol{\lambda}_{p}$** column and using that sum in the reliability function gives us

$R(t) = e^{-(17.6092)*10^{-6}*365.25*24} = 85.7\%$

Thus, there is an 85.7% chance that the circuit will not fail if it were to be left on during a one-year period.

Sources

C. Davis, "Circuit Reliability Theory."

"Simple Combination Lock: Digital Integrated Circuits: Electronics Textbook," *All About Circuits*. [Online]. Available: https://www.allaboutcircuits.com/textbook/experiments/chpt-7/simple-combination-lock/. [Accessed: 25-Mar-2021].