Digital Security Lock System

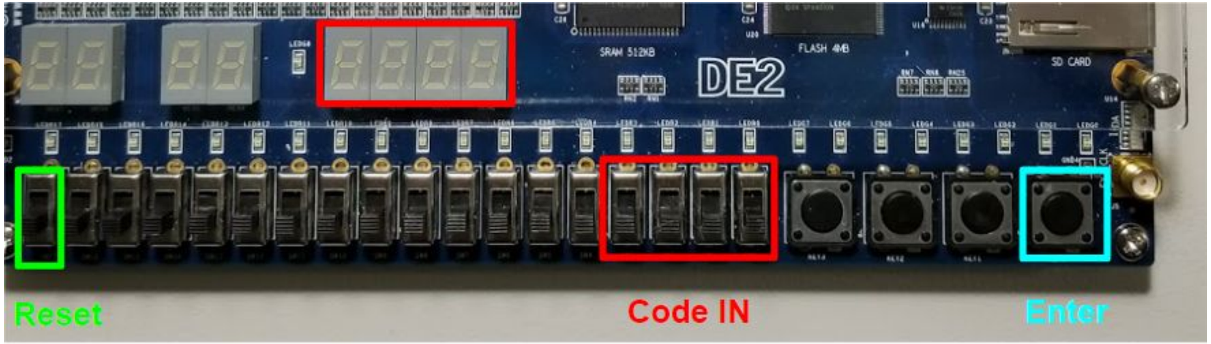
Chris Barr Jett Guerrero

Function

- A security lock that requires a password code to unlock a system. The locking system has a digital lock display to indicate the current condition, locked or unlocked. The password code is resettable if one decides to change it.

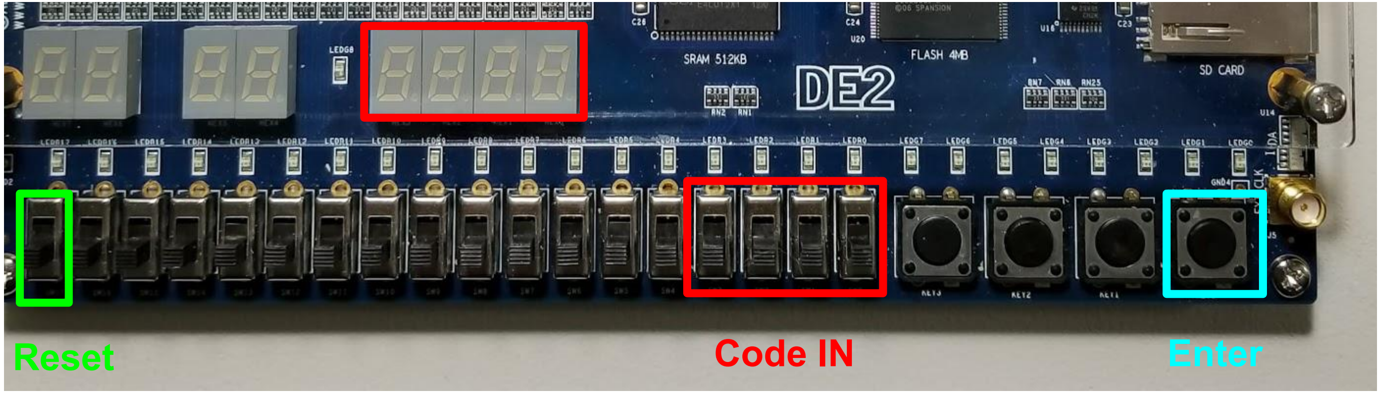
Operation Process

- Passcode is to be entered using four switches on the DE2 board.
- The passcode values (0 or 1) is displayed on the seven segment display.
- After the password code is entered, the right most push button must be pressed to unlock the system.
- If password is valid, the digital lock opens and turns green. If invalid, the lock remains red and locked.
- If one wishes to change the passcode, one must first enter the valid password then switch on the left most switch to enter a new passcode. In this mode, digital lock turns yellow.



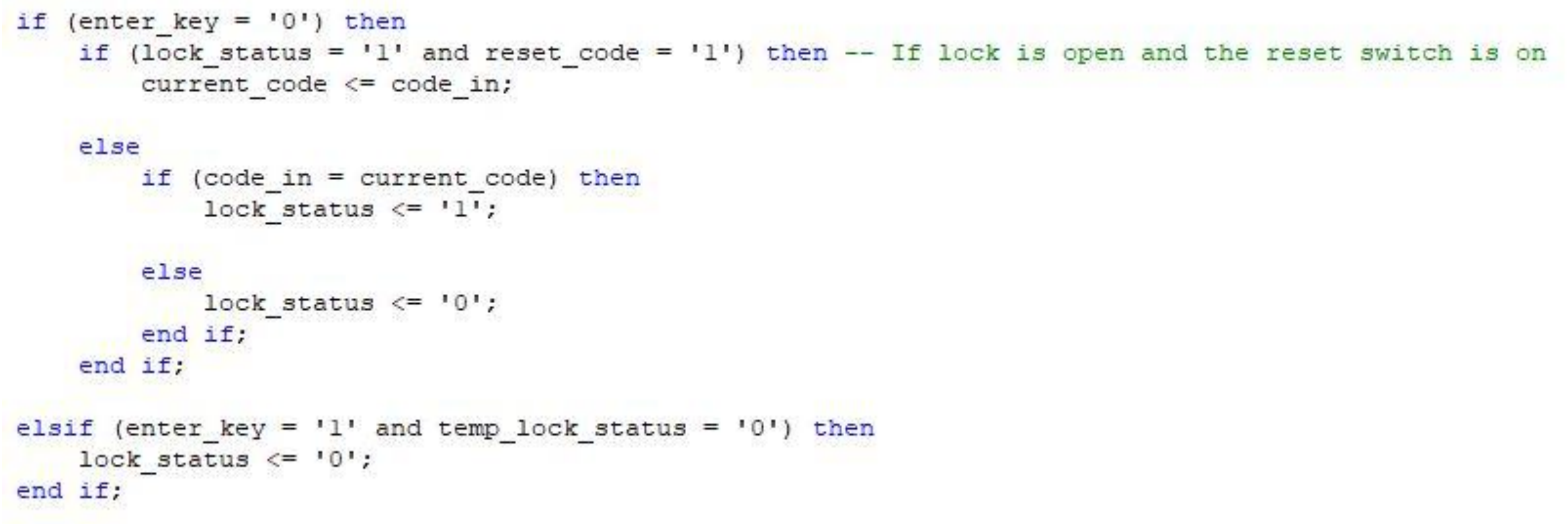
Implementation

- VHDL language
  - Sequential code using two onboard clocks (25MHz & 50MHz) followed by pin-map.
- Switches & Push Button
  - When switch is switched to high(1), the value is stored as a current password code
  - If-else statements to check if entered password matches the current password



Implementation

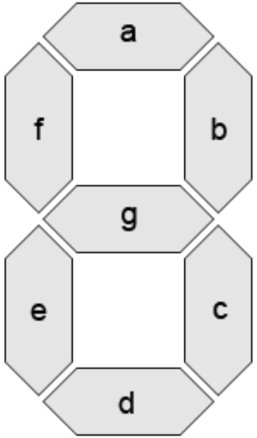
- Lock Logic

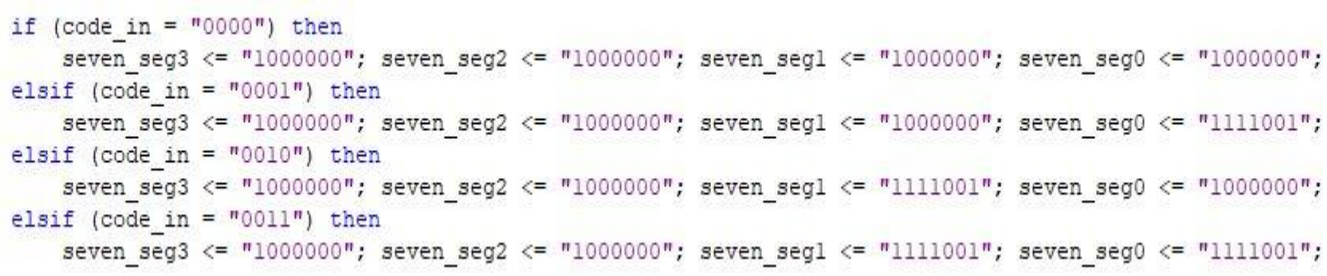


```
if (enter_key = '0') then
    if (lock_status = '1' and reset_code = '1') then -- If lock is open and the reset switch is on
        current_code <= code_in;

    else
        if (code_in = current_code) then
            lock_status <= '1';

        else
            lock_status <= '0';
        end if;
    end if;

elsif (enter_key = '1' and temp_lock_status = '0') then
    lock_status <= '0';
end if;
```

Implementation

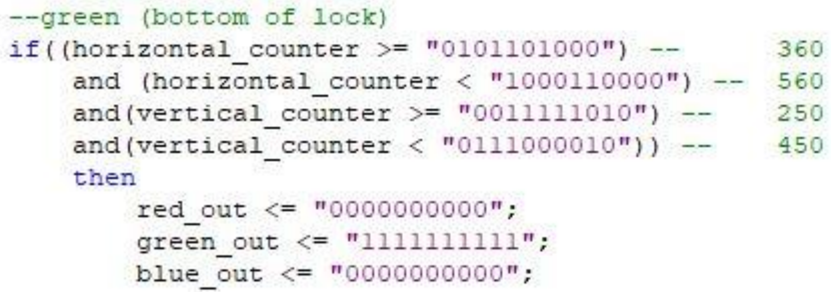
- Seven Segment Display
  - Used Hex3-Hex0
  - If-else statements to link switches to the seven segment display
  - Seven segment controlled by 7 bits. When a segment needs to be turned on, a “0” is placed in that bit position. Otherwise “1” for off.
    - Ex: Display a “1” = “GFEDCBA” --- “1111001”



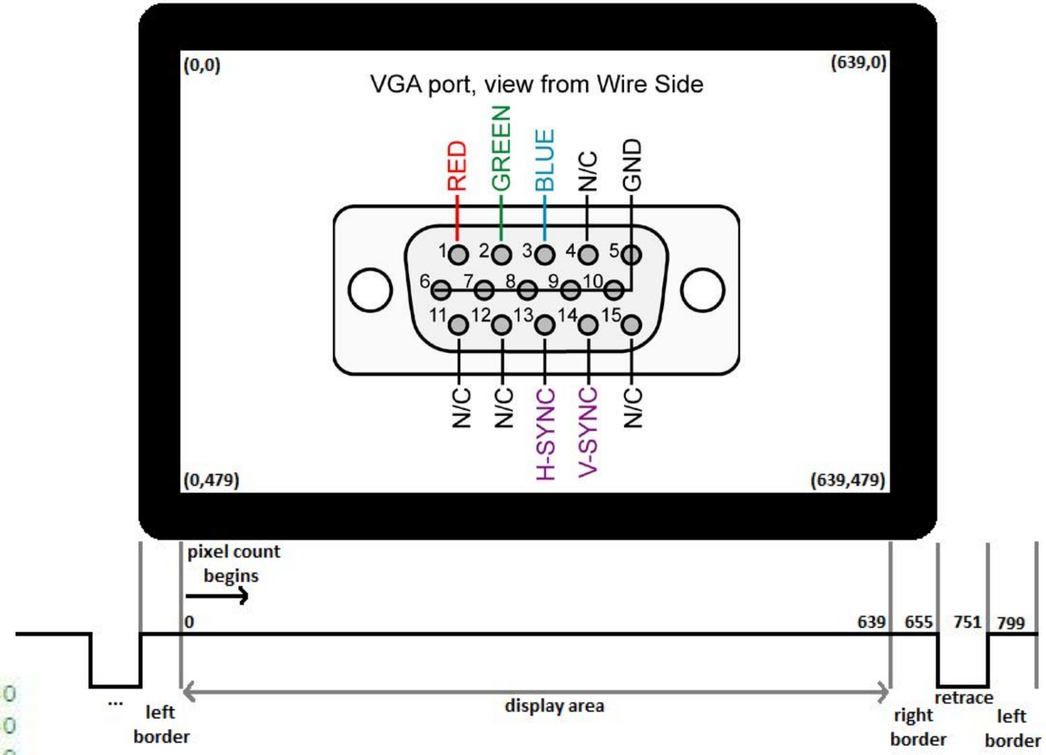

```
if (code_in = "0000") then
    seven_seg3 <= "1000000"; seven_seg2 <= "1000000"; seven_seg1 <= "1000000"; seven_seg0 <= "1000000";
elsif (code_in = "0001") then
    seven_seg3 <= "1000000"; seven_seg2 <= "1000000"; seven_seg1 <= "1000000"; seven_seg0 <= "1111001";
elsif (code_in = "0010") then
    seven_seg3 <= "1000000"; seven_seg2 <= "1000000"; seven_seg1 <= "1111001"; seven_seg0 <= "1000000";
elsif (code_in = "0011") then
    seven_seg3 <= "1000000"; seven_seg2 <= "1000000"; seven_seg1 <= "1111001"; seven_seg0 <= "1111001";
```

Implementation

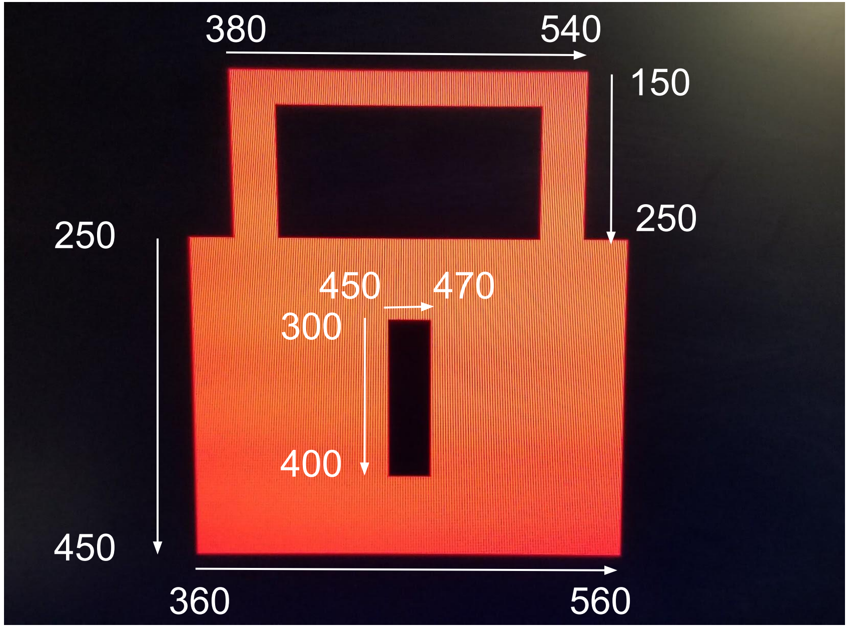
- VGA Display
  - Resolution of 640x480
  - VGA pins assigned to RGB pin assignments
  - Generated shapes by calculating the corner pixel coordinates of each shape

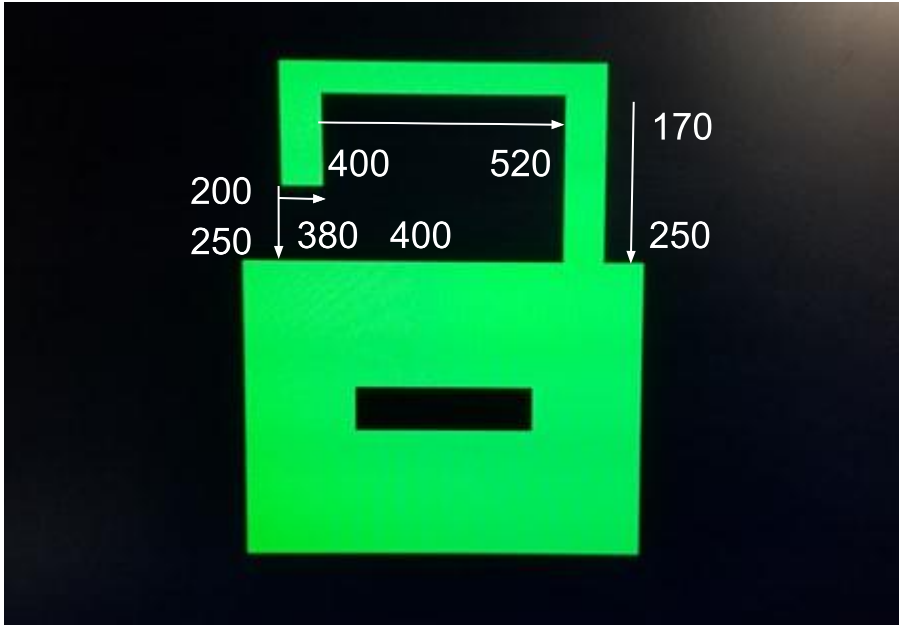


```
--green (bottom of lock)
if((horizontal_counter >= "0101101000") --      360
    and (horizontal_counter < "1000110000") --  560
    and(vertical_counter >= "0011111010") --    250
    and(vertical_counter < "0111000010")) --    450
    then
        red_out <= "0000000000";
        green_out <= "1111111111";
        blue_out <= "0000000000";
```



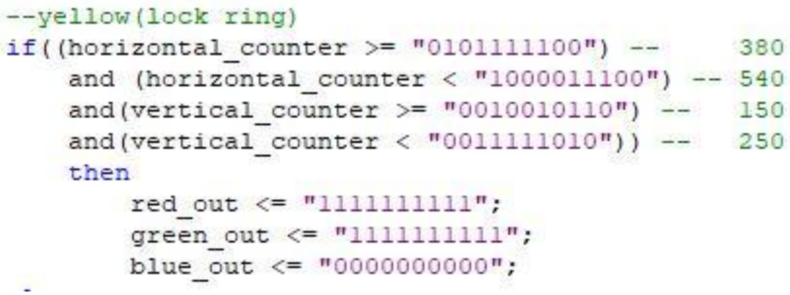
Implementation





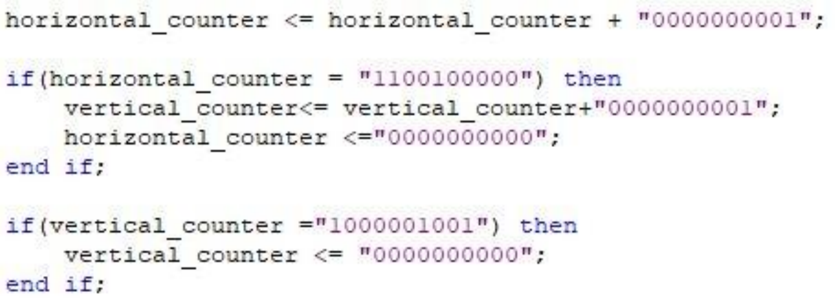
Implementation

- Assigning color by outputting 10 bits of “1” or “0” to the RGB output



```
--yellow(lock ring)
if((horizontal_counter >= "0101111100") --     380
    and (horizontal_counter < "1000011100") -- 540
    and(vertical_counter >= "0010010110") --   150
    and(vertical_counter < "0011111010")) --   250
    then
        red_out <= "1111111111";
        green_out <= "1111111111";
        blue_out <= "0000000000";
```

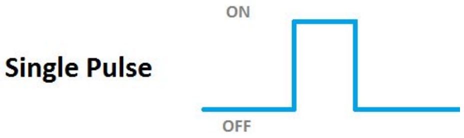
- If-else statements to output the image on the screen. Each pixel is checked with every screen refresh.

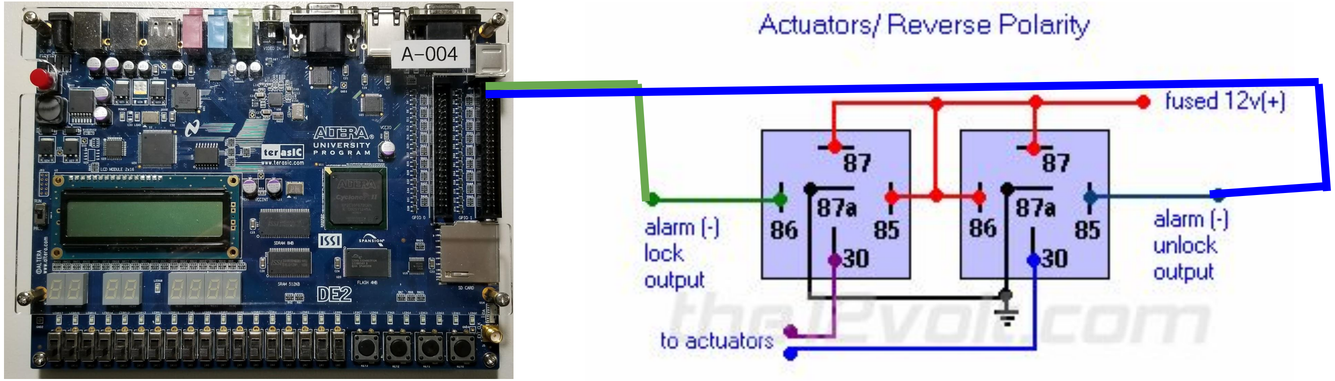


```
horizontal_counter <= horizontal_counter + "0000000001";

if(horizontal_counter = "1100100000") then
    vertical_counter<= vertical_counter+"0000000001";
    horizontal_counter <="0000000000";
end if;

if(vertical_counter ="1000001001") then
    vertical_counter <= "0000000000";
end if;
```

Implementation

- Relays and Actuator
  - ○ 3.3v sent to one of two GPIO pins for when status is locked or unlocked
  - ○ Two relay switches are needed to open and close the actuator. Each controlled by one GPIO pin

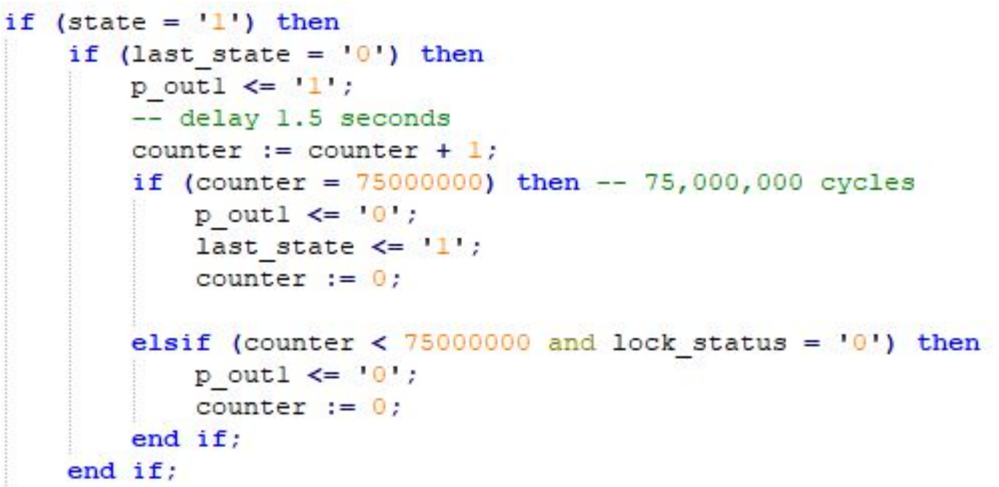




Implementation

If it’s unlocked and the last state of the lock was locked, output 3.3V for 1.5 sec.

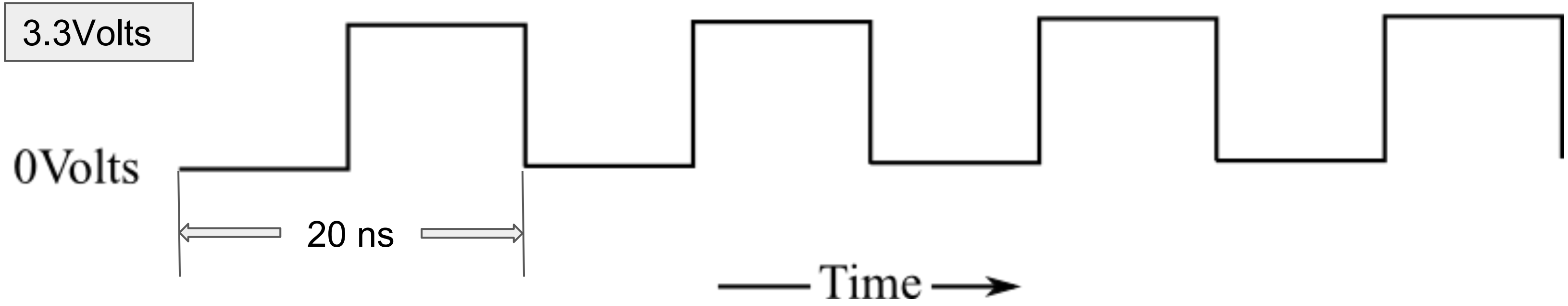
Else, if the lock goes from unlocked to locked before 1.5 sec. do not output 3.3V.



```
if (state = '1') then
    if (last_state = '0') then
        p_out1 <= '1';
        -- delay 1.5 seconds
        counter := counter + 1;
        if (counter = 75000000) then -- 75,000,000 cycles
            p_out1 <= '0';
            last_state <= '1';
            counter := 0;

        elsif (counter < 75000000 and lock_status = '0') then
            p_out1 <= '0';
            counter := 0;
        end if;
    end if;
```

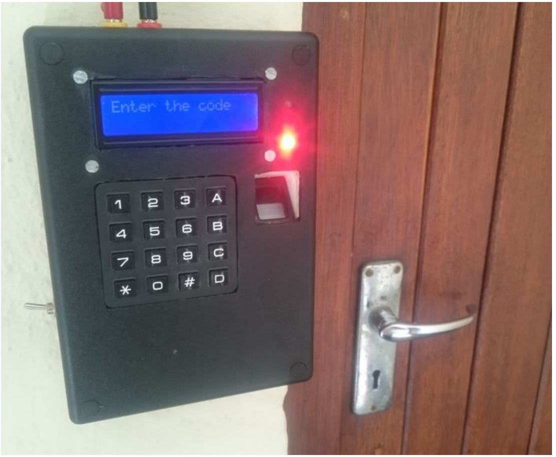
Implementation

T = (1/f)



Applications

- House doors
- Car doors
- Safes
- Office doors
- Lockers



Encountered Problems

- When trying to create a new password code, lock automatically locks
- Input is checked with the rising edge of the 50MHz clock which sometimes causes input errors.
- Outputting a 3.3v signal from the GPIO pins.
- Setting the counter to count up to 3 seconds with the frequency of the 50MHz clock.
- GPIO output voltage signal is too low. Decided to increase voltage using an external battery in series which caused issues with the two relays.
- At the moment, actuator lock can only open but not close.

Roles

- Chris
  - Programmed lock logic - setting password, resettable password
  - Implemented switches and pushbutton
  - Coded to output a signal logic out of the GPIO
- Jett
  - Programmed and designed the lock screens
  - Seven segment display
  - Relay and actuator circuit

Thank you