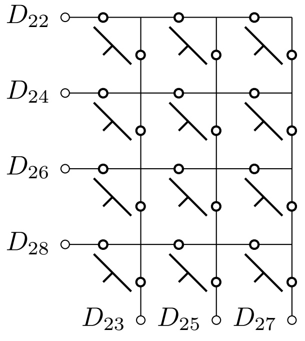
Lab 7
Displays

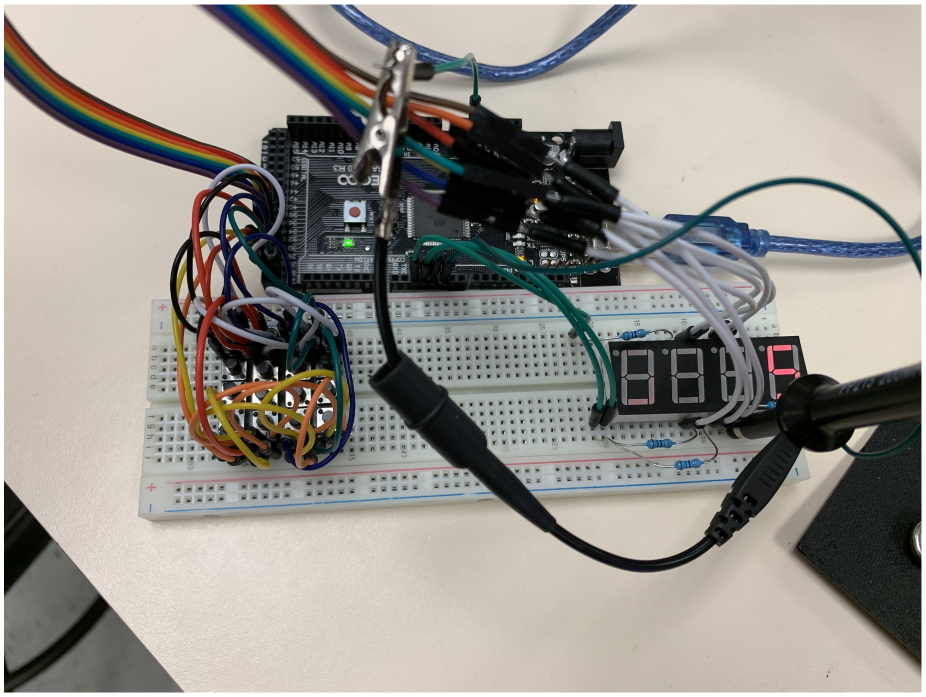
Keaton Clark

October 25, 2022

Circuit

I did not purchase the kit as I have *most* of the parts required for the class. One of the parts I did not have was the button matrix so I needed to build one myself. Here is the schematic.





Code

The important logical portion of main.c, everything above is just initializing the button matrix and the 7 segment display. Looking back at my code I see it actually gets stuck at the first while loop (updating the display the whole time) until a button is pressed then it loops backwards from i=3, only decrementing when a key is pressed and write the value of the keypress to the ith eleement of a 4 element char array. Write the char array to the display and then get stuck at the while loop updating the display until the key is unpressed and then another key is later pressed.
Once it reads four chars it breaks out of the loop and goes into a psuedo delay loop updating the whole time so that we can see the keys we just input. Then clear the char array, write the clear array to the display, update, and do it all again.

```c
    for(;;) {
        for (int i = 3; i >= 0;) {
            key_press = read_matrix(matrix);
            if (key_press != 0) {
                display[i] = key_press;
                i--;
            }
            four_dig_seven_seg_write(&sevenseg, display);
            while (key_press == read_matrix(matrix)) { // Wait for the keypress to be un-pressed. Update while delaying
                four_dig_seven_seg_update(sevenseg);
            }
            _delay_ms(10); // Little delay because reasons
        }
        for (int i = 0; i < 500; ++i) { // Delay but continue to update while delaying
            four_dig_seven_seg_update(sevenseg);
        }
        memset(display, 0, 4);
        four_dig_seven_seg_write(&sevenseg, display);
        four_dig_seven_seg_update(sevenseg);
    }
```

The matrix code is pretty simple. In the init set all rows to outputs and pull them high, and set all the columns to inputs and activate their internal pullups. Then to read a certain button on the matrix set its corresponding row to low and read the value on its corresponding column.

```c
#include "matrix.h"

matrix_t init_matrix(uint8_t rows, pin_t* row_pins,
    uint8_t cols, pin_t* col_pins, char* lookup_table) {
    for (int i = 0; i < rows; ++i) {
        pin_mode(row_pins[i], OUTPUT);
        write_pin(row_pins[i], HIGH); // Start polling
    }
    for (int i = 0; i < cols; ++i) {
        pin_mode(col_pins[i], INPUT);
        write_pin(col_pins[i], HIGH); // Activate
    internal Pull-ups
    }
    return (matrix_t) {
        .rows = rows,
        .row_pins = row_pins,
        .cols = cols,
        .col_pins = col_pins,
        .lookup_table = lookup_table
    };
}

char read_matrix(matrix_t matrix) {
    for (int i = 0; i < matrix.rows; ++i) {
        write_pin(matrix.row_pins[i], LOW); // Look at
    pin

        for (int j = 0; j < matrix.cols; ++j) {
            if (!read_pin(matrix.col_pins[j])) {
                write_pin(matrix.row_pins[i], HIGH); //
    Reset pin to high
                return matrix.lookup_table[(i * matrix.
    cols) + j];
            }
        }

        write_pin(matrix.row_pins[i], HIGH); // Reset pin
     to high
    }
    return 0;
}
```

The seven segment code is a similar idea as the matrix. Set all segment and digit pins as outputs and write all of the digit pins high. Then when you want to write to a digit, pull that one low and use the lookup table to receive an array with each element being the state of the segment in the corresponding number you used in the table. Then loop through the segments and write to them the state that the table tells you to. In order to generate this table I just used trial and error writing digits to the display.

```c
#include "sevenseg.h"

uint8_t seven_seg_lookup[10][7] = {
    [0] = {0, 1, 1, 1, 1, 1, 1},
    [1] = {0, 1, 0, 0, 0, 1, 0},
    [2] = {1, 0, 1, 1, 0, 1, 1},
    [3] = {1, 1, 1, 0, 0, 1, 1},
    [4] = {1, 1, 0, 0, 1, 1, 0},
    [5] = {1, 1, 1, 0, 1, 0, 1},
    [6] = {1, 1, 1, 1, 1, 0, 0},
    [7] = {0, 1, 0, 0, 0, 1, 1},
    [8] = {1, 1, 1, 1, 1, 1, 1},
    [9] = {1, 1, 0, 0, 1, 1, 1},
};

four_dig_seven_seg_t init_four_dig_seven_seg(pin_t *dig,
    pin_t *seg) {
    for (int i = 0; i < 4; ++i) {
        pin_mode(dig[i], OUTPUT);
        write_pin(dig[i], HIGH);
    }
    write_pin(dig[0], LOW);
    for (int i = 0; i < 7; ++i) {
        pin_mode(seg[i], OUTPUT);
        write_pin(seg[i], LOW);
    }
    return (four_dig_seven_seg_t) {
        .dig = dig,
        .seg = seg,
    };
}
void four_dig_seven_seg_write(four_dig_seven_seg_t *
    display, char content[4]) {
    for (int i = 0; i < 4; ++i) {
        display->dis[i] = content[i];
    }
}
void four_dig_seven_seg_update(four_dig_seven_seg_t
    display) {
     for (int i = 0; i < 4; ++i) {
          write_pin(display.dig[i], LOW);
          if (display.dis[i] >= 48) {
               for (int j = 0; j < 7; ++j) {
                    write_pin(display.seg[j],
    seven_seg_lookup[display.dis[i] - 48][j]);
               }
          } else {
               for (int j = 0; j < 7; ++j) {
                    write_pin(display.seg[j], LOW);
               }
          }
          _delay_ms(1);
          write_pin(display.dig[i], HIGH);
     }
}
```

Question

- One of the leads of the seven segment display (I believe this is one of the segment leads) is showing a square wave. This is because the Arduino cannot write to each digit at all times so it cycles through them and there is a small delay in between each digit, although unseen to the naked eye. This segment seems to be turned on for each of the digits, otherwise there would be a corresponding gap in the square wave.