Project Tank

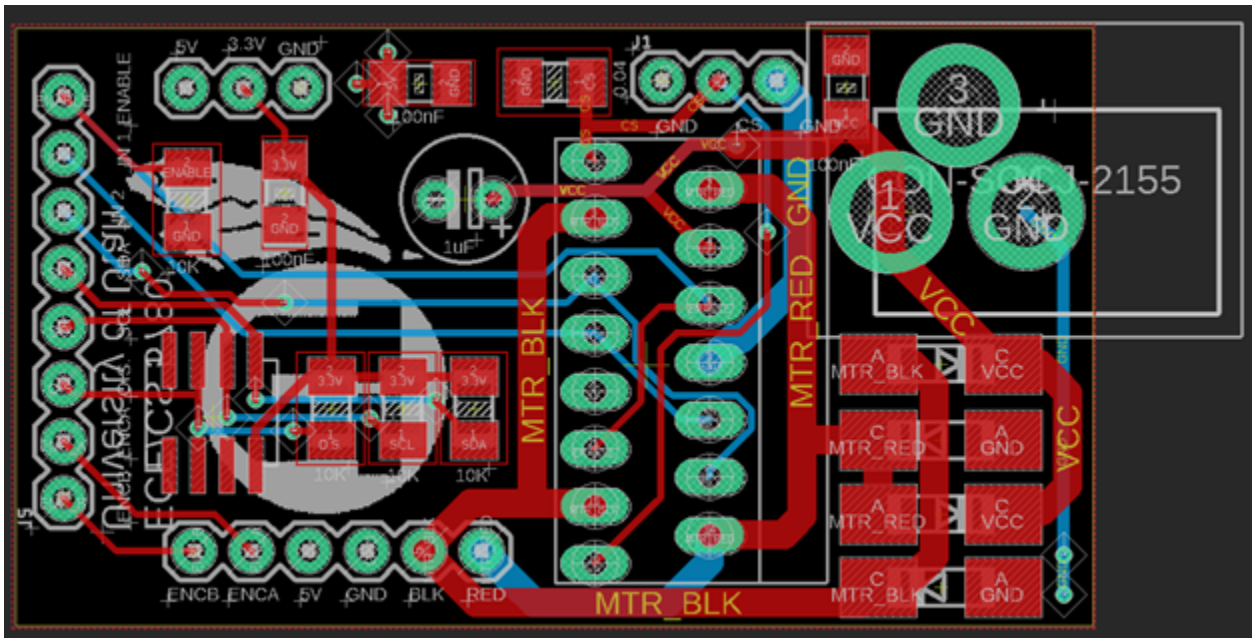
Brian Burton, Matthew Hunsaker, Kashish Singh

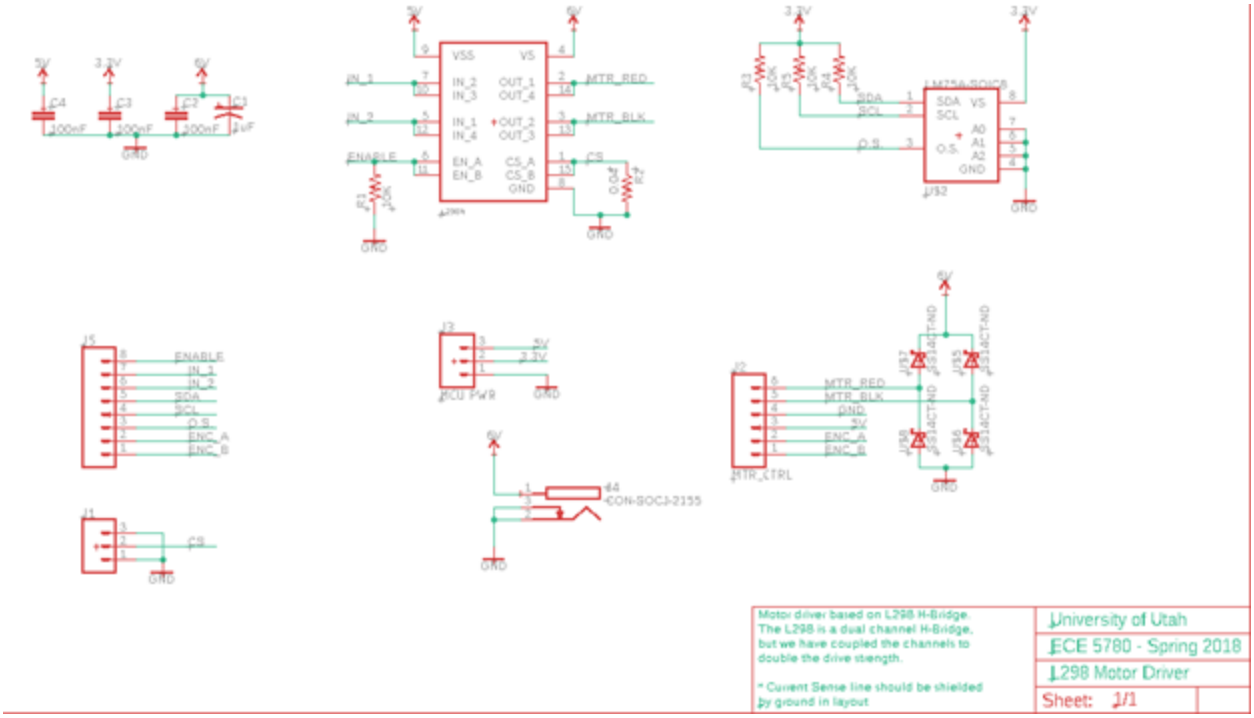
Introduction

The project which has been nicknamed “Project Tank” was the idea of taking the homework assignments for the 5V DC Motor design and putting it into a functioning device. Essentially our project is able to move on the ground and turn by utilizing 2 motors, treads, a battery pack for the 6V output, the STM32F0 board, and a 3D printed chassis to hold all the components.

Setup

The project utilizes the PCB board designed for the class. The one ordered was based on the homework solution to avoid any potential issues with the board design or sauntering issues since it’s guaranteed to work with the motor as long as there isn’t user error. Since our design uses 2 motors we needed 2 PCB boards that were fully sauntered.





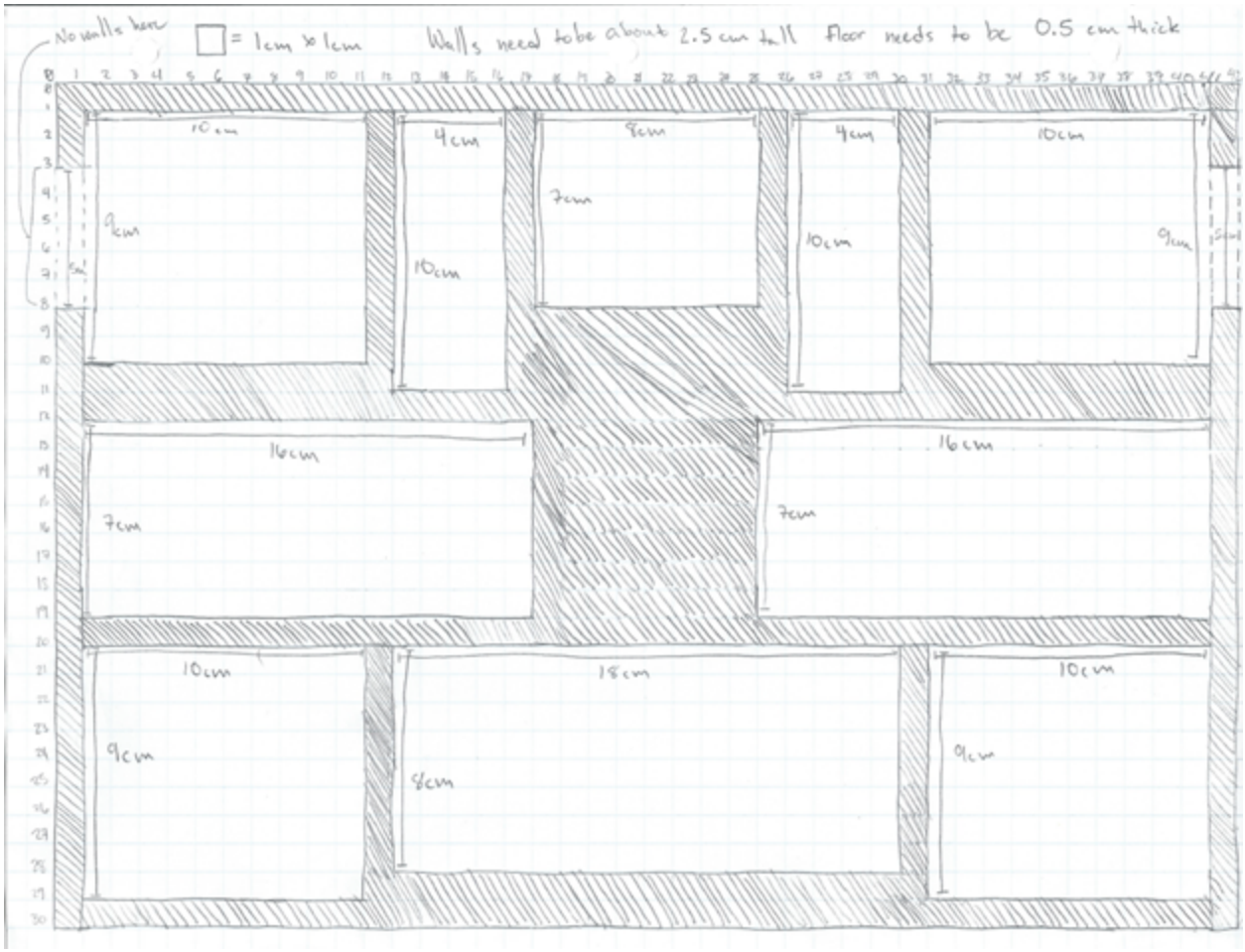
The next part after the PCB design was the motors themselves. Unfortunately, the UofU stockroom in the Merrill Engineering Building did not have 2 motors nor could we use the one they had since we would constantly be testing and connecting the motors. So, we had to purchase 2 from PLU (https://www.pololu.com/product/2824) for a total of $103 without tax and shipping ($140 with). Other parts were then acquired such as, synchronous wheels with pulleys, tread belts to move the wheels, and a battery holder with 5 AA batteries to avoid needing a 6V adapter and having the tank be dependent on a wall socket. The STM32F0 board was used to provide the code and other power sources for the motors and itself is powered by a portable battery supply that the user has.

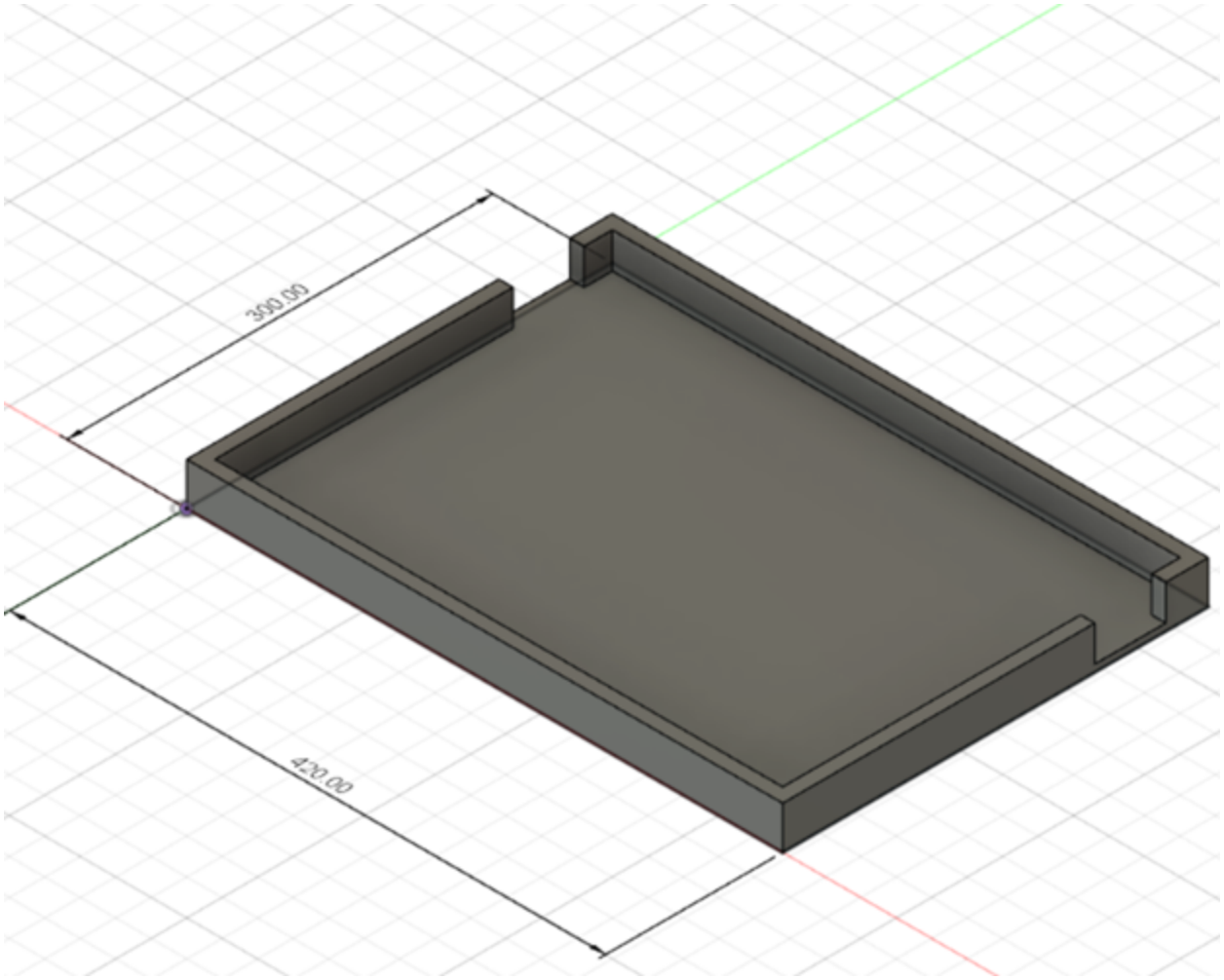
Bill of Materials

Name	Description and Store	Quantity	Individual Price	Total
Pololu Gearmotor	https://www.pololu.com/product/4753 - Polulu	2	$51.95	$103.9
Zeberoxyz 6.35mm Pully	20&60 Teeth bore aluminum timing pully w/ 200mm width belts - Amazon	1	$12.89	$12.89

DC Power Splitter Cable	5.5mm x 2.1mm 32cm DC 1 Male to 2 Female - Amazon	1	$7.49	$7.49
6V Battery Holder	ZRM&E with 5.5x2.1 barrel jack connector - Amazon	1	$6.99	$6.99
Zeberoxyz Closed Belt	Various sizes 3D printer closed belt - Amazon	2	$12.89	$25.78
3D printed chassis	UofU Marriot Library 3D printing services	1	Based on grams	$27.26
			TOTAL:	$184.31

The 3D printed chassis had to be designed for all the different components that would be placed as well as any location that needed to be drilled to secure the pullies, motors, and belts. First a sketch and diagram was made on paper to give the dimensions and placements. Then using Fusion 360 the chassis was designed based off of the sketch. From there the chassis was printed with the University of Utah printing services in the Marriot Library.


No walls here
□ = 1cm × 1cm
Walls need to be about 2.5 cm tall
Floor needs to be 0.5 cm thick
10 cm
4cm
8cm
4cm
10cm
4cm
10cm
7cm
10cm
9cm
16cm
7cm
16cm
7cm
10cm
9cm
18cm
8cm
10cm
9cm



From there, the finished chassis was drilled with the different components secured to the base and the wiring was done to properly connect the 2 motors to the 2 PCB boards as well as the battery supply and the STM32F0 Discovery board.

Software

With the hardware acquisition and setup completed and us using modeling techniques from the lab for designing the different components and connections, next came the software. Since the STM32F0 board was the only device used for this task, the code was in C with the use of CubeMX and Kiel Uvision5 software. The software consists of several functions in addition to the main function. These functions include some set up for the TIM2 and TIM3 on the STM board as well as set up for the board's user button and the LEDs. TIM3 is set to trigger its interrupt with a frequency of 8MHz. The TIM3 interrupt acts as a debouncer for the user button and inverts the unsigned 8 bit start_stop variable whenever the button is pressed. The TIM2 timer operates at a frequency of 4Hz and the TIM2 interrupt iterates the distance_time variable. The distance_time variable keeps track of how many times the TIM2 interrupt is triggered and

currently has a max value of 240, though this can be changed as needed. Since TIM2 operates at a frequency of 4Hz that means that distance_timer will be iterated 4 times for every second that passes and the 240 limit would be a 1 minute length of time.

The only other functions in the code are the directional functions move_forward, move_reverse, turn_right, and turn_left with the move_forward button displayed below.. All of these functions have the same configuration with the only thing that changes between them is which LED pins receive power. The format of these functions consists of three major parts. The first part is the TIM2 initialization function is called to start TIM2 ticking at 4Hz. The second part of the function is a while loop that supplies power to the LEDs and their pins for the specific direction until the distance_timer reaches the total travel time in a single direction indicated by the distance_change_time variable. The distance_change_time variable stores the number of seconds the vehicle will move in a given direction so the distance_timer variable must be divided by 4 to get the current time in seconds. The third part of the movement functions after the while loop disables the TIM2 clock, disables the power to the LED pins and resets the distance_timer variable to 0. Since the directional pins of the motors are connected to the LED output pins the board LEDs can be used to determine the direction the vehicle is currently moving with the red and orange LEDs for forward, blue and green for reverse, red and green for turning right, and blue and orange for turning left. The functions are then called in the main loop in whatever order the user desires for automated movement and the distance_change_variable can be changed within the main loop to change the distance traveled.

```
/*
 * This function causes the vehicle to move forward. This function is tested
 * using the LEDs.
 */
void move_forward(void) {
  TIM2_init();
  while(distance_change_time > (distance_timer / 4)) {
    // Turn on the red & orange LEDs while moving forward.
    GPIOC->BSRR |= (1 << 6) | (1 << 8);
  }
  // Turn off the TIM2 timer.
  TIM2->CR1 &= ~(0xFFFE);
  // Turn off the LEDs when done.
  GPIOC->BSRR |= (1 << 22) | (1 << 23) | (1 << 24) | (1 << 25);
  // Reset the value of the distance_timer
  distance_timer = 0;
}
```

The main function, shown below, contains function calls for each direction within the an if statement. The if statement will not start the vehicle moving until the user button is pressed. Once pressed the vehicle will move until the end of the sequence when the start_stop variable is set to zero again. If the user button was pressed while the vehicle was in the middle of moving

that movement would continue until the vehicle loses power. This allows the user to set the vehicle to a direction in the middle of moving if the need arises.

```
/*
 * Main program
 */
int main(int argc, char* argv[]) {
  HAL_Init();
  GPIO_init();
  LED_init();
  button_init();

  while(1) {
    if(start_stop != 0) {
      move_forward();
      move_reverse();
      turn_right();
      turn_left();
      start_stop = 0;
    } else {
      // Turn off the TIM2 timer.
      TIM2->CR1 &= ~(0xFFFE);
      // Turn off the LEDs when done.
      GPIOC->BSRR |= (1 << 22) | (1 << 23) | (1 << 24) | (1 << 25);
      // Reset the value of the distance_timer
      distance_timer = 0;
    }
  }
}
```

Improvements

If there was more time there could be various other improvements added to the design. One of the things discussed was potentially having the project be more functional for everyday use rather than just having a moveable block. The idea was presented to turn the initial concept into a “portable table” that could potentially hold lightweight items while still being able to move around. However, the design was already too far along to adjust to this new idea.

Another aspect that could be improved is adding more customization or control over the device from the user. Such as using one of the STM32F0 buttons in order to start and stop the device instead of just having the device run infinitely in one direction. Potentially even using the joystick on the STM board to possibly manually change directions instead of needing to pre-program the steps or add it to be able to interrupt the pre-programmed steps.

Conclusion

From this final project a lot was learned. From how to properly model and splice designs for 3D printing as well as PCB board making and soldering, to connections and gear movements with belts and pulleys as well as the creation of software using timings and interrupts for our design. Overall, it has been an interesting way to demonstrate what we've learned and how embedded systems can be used to create and control devices for everyday use.