Embedded Systems - Final Project

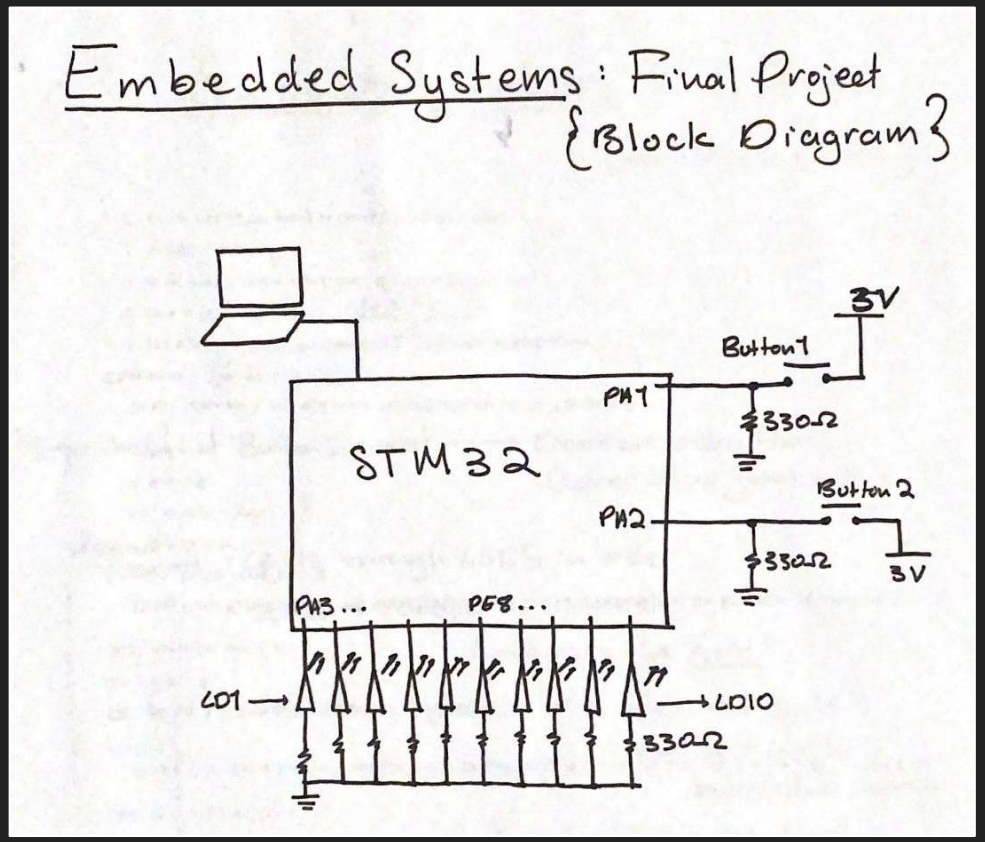
Josh Mejia

the idea

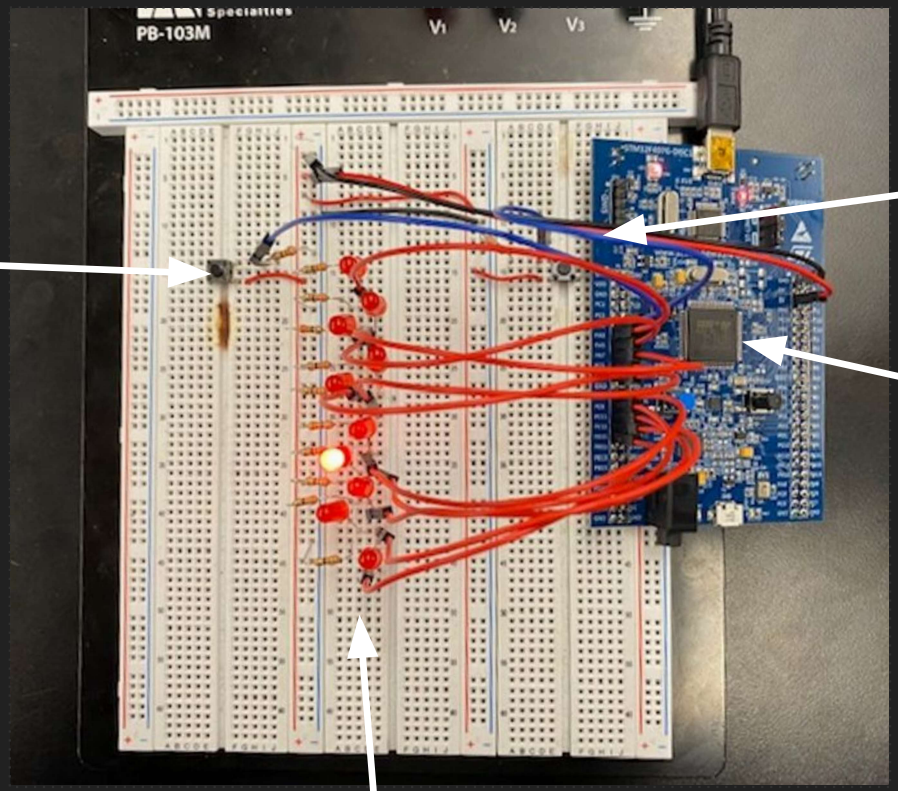
Remember this thing?

the diagrams

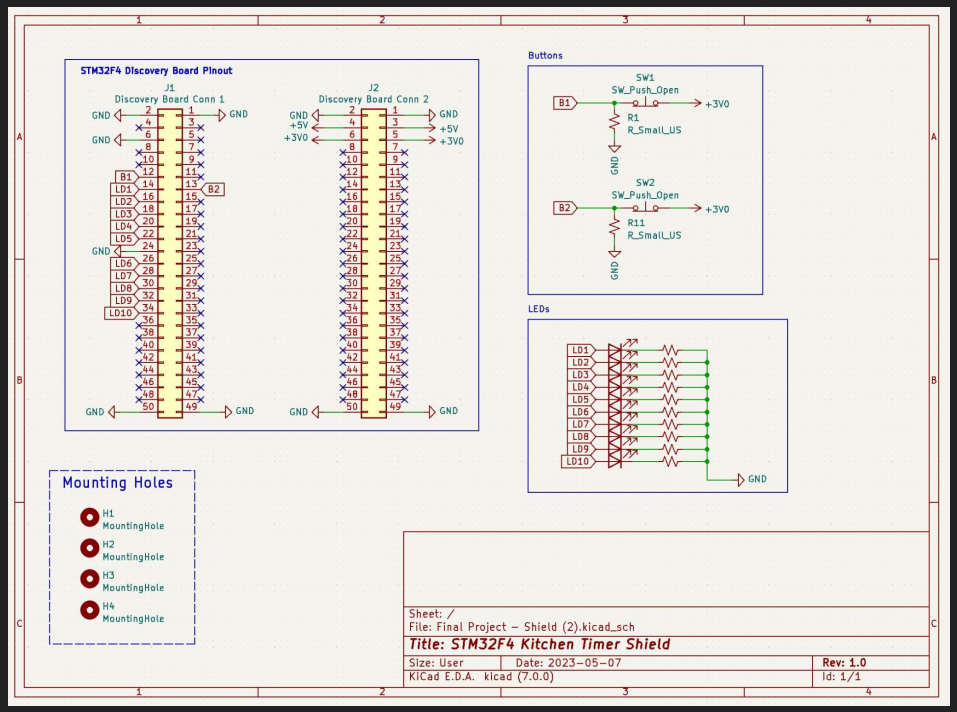
prototype:



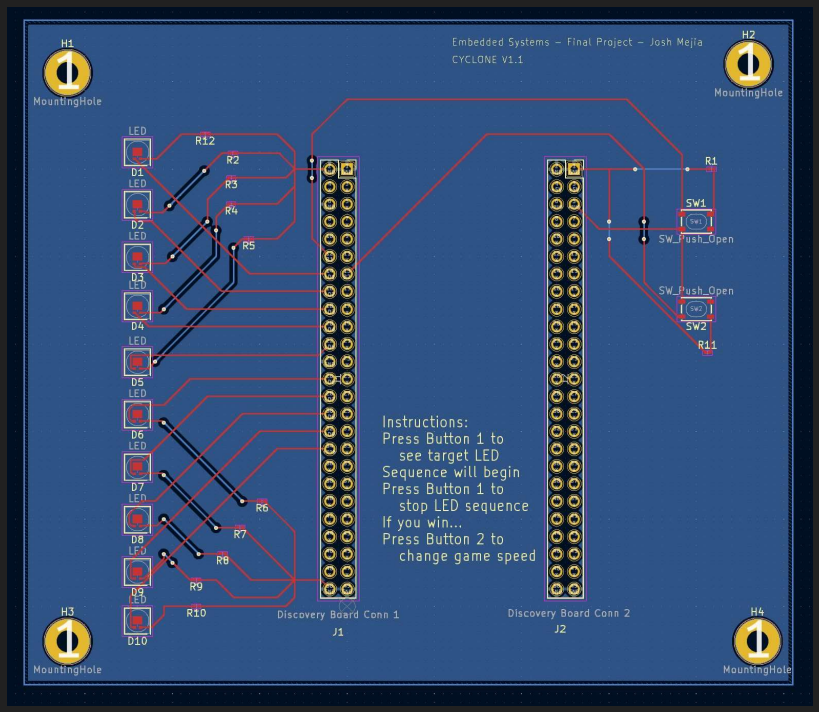
the prototype



the pcb design

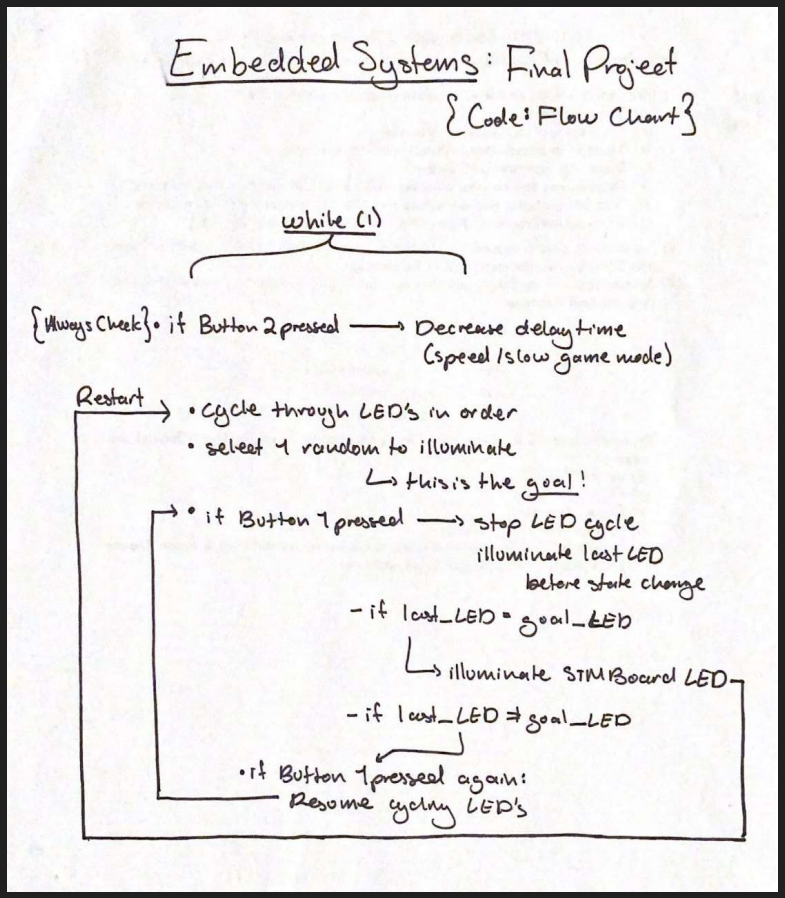


schematic



layout

the code



while begins

if (button 2)

if (button 1)

LED1 Cycle

If (button 1)

LED2 Cycle

If (button 1)

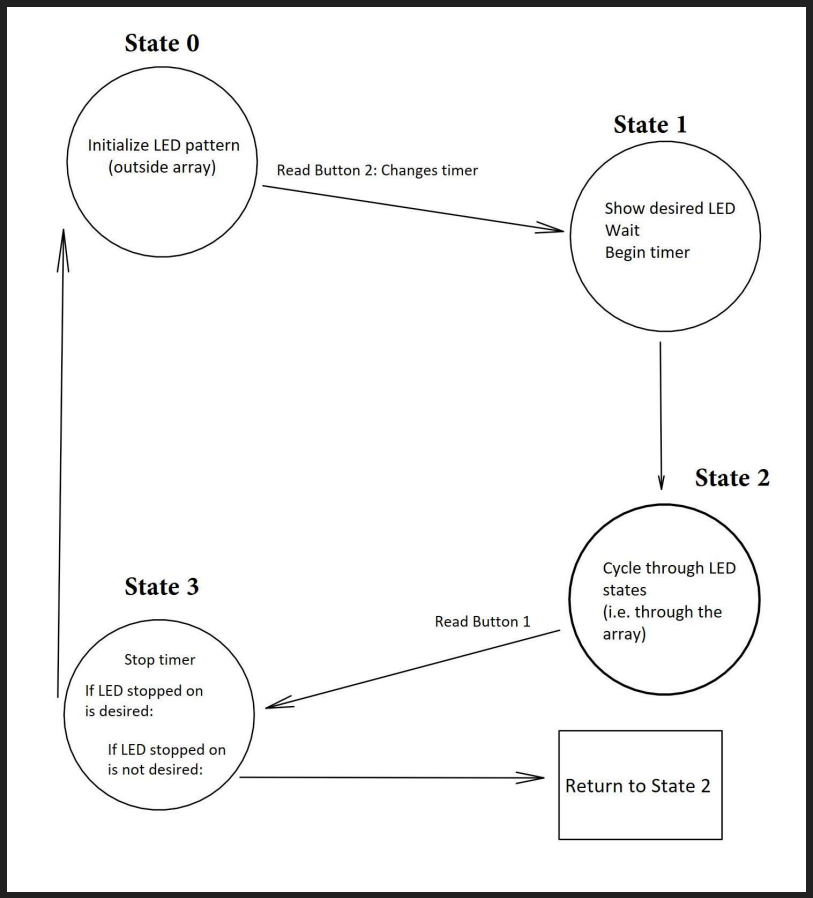
LED3 Cycle

If (button 1)

. . .

```
/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
    // button 2 - change speed
    if (gISR_Bool2){
        wait_time = 100;
    } else {
        wait_time = 50;
    }
    if (gISR_Flag2 == 1){
            gISR_Flag2 = 0;
            gISR_Bool2 = !gISR_Bool2;
        }

    // button 1 - start/stop
    if (gISR_Flag1 == 1){
            gISR_Flag1 = 0;
            gISR_Bool1 = !gISR_Bool1;
        }

    ////////////////////////////////////////////////////
    // LED Pattern Begin
    if (gISR_Bool1){
        HAL_GPIO_WritePin(GPIOA, LD10_Pin, GPIO_PIN_RESET);
        HAL_Delay(wait_time);
        HAL_GPIO_WritePin(GPIOE, LD1_Pin, GPIO_PIN_SET);
        HAL_Delay(wait_time);
    }
    if (gISR_Flag1 == 1){
            gISR_Flag1 = 0;
            gISR_Bool1 = !gISR_Bool1;
        }

    // button 1 - start/stop
    if (gISR_Bool1){
        HAL_GPIO_WritePin(GPIOE, LD1_Pin, GPIO_PIN_RESET);
        HAL_Delay(wait_time);
        HAL_GPIO_WritePin(GPIOE, LD2_Pin, GPIO_PIN_SET);
        HAL_Delay(wait_time);
    }
    if (gISR_Flag1 == 1){
            gISR_Flag1 = 0;
            gISR_Bool1 = !gISR_Bool1;
        }

    if (gISR_Bool1){
            HAL_GPIO_WritePin(GPIOE, LD2_Pin, GPIO_PIN_RESET);
            HAL_Delay(wait_time);
            HAL_GPIO_WritePin(GPIOE, LD3_Pin, GPIO_PIN_SET);
            HAL_Delay(wait_time);
        }
    if (gISR_Flag1 == 1){
            gISR_Flag1 = 0;
            gISR_Bool1 = !gISR_Bool1;
        }


    if (gISR_Bool1){
            HAL_GPIO_WritePin(GPIOE, LD3_Pin, GPIO_PIN_RESET);
            HAL_Delay(wait_time);
            HAL_GPIO_WritePin(GPIOE, LD4_Pin, GPIO_PIN_SET);
            HAL_Delay(wait_time);
        }
    if (gISR_Flag1 == 1){
            gISR_Flag1 = 0;
            gISR_Bool1 = !gISR_Bool1;
        }
```

the code part II




```c
/* USER CODE BEGIN 4 */
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim)
{
    if(htim == &htim1)
    {
        gISR_Flag_1 = 1;
    }
}
/* USER CODE END 4 */
```

```c
/* Private variables ---------
TIM_HandleTypeDef htim1;

/* USER CODE BEGIN PV */
char gState = 0;
char gISR_Flag_1 = 0;
/* USER CODE END PV */

/* Private function prototypes --
void SystemClock_Config(void);
static void MX_GPIO_Init(void);
static void MX_TIM1_Init(void);
/* USER CODE BEGIN PFP */
```

```c
HAL_TIM_Base_Start_IT(&htim1);

/* USER CODE END 2 */

/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
    if(gISR_Flag_1 == 1){
        gISR_Flag_1 =0;

        switch(gState)
        {
        // STATE 0 - Init State
        case 0:
            HAL_GPIO_WritePin(GPIOE, GPIO_PIN_15, SET);
            //HAL_Delay(1000);
            gState = 1;  // go to next state
            break;

        case 1:
          HAL_GPIO_WritePin(GPIOE, GPIO_PIN_15, RESET);
          //HAL_Delay(1000);
          gState = 0;  // go to next state
          break;

        default:
            break;

        }
    }
  /* USER CODE END WHILE */
```

the errors

- designated LED
  - the program would illuminate a selected LED, but would not register it as a target, so it was not comparable to the current state of the user in the LED cycle
- timer
  - could not get second button to change the rate
- button interrupt
  - the first button, using the new code, would register the button switch only in between stages (which was in between clock cycles). the idea is for it to interrupt the signal at any given point, so this behavior was not ideal

the reflection

- state machine
  - next time, i would like to avoid hard-coding this section of the code, especially for transitioning between states. the way to avoid this would be to write a function that rotates through calling the GPIO_WritePin function on all LED’s
- button interrupts
  - the only way i was able to make the button successfully interrupt the signal was with a constant checking, instead of having a flag that would be changed at any point (asynchronously).

the end