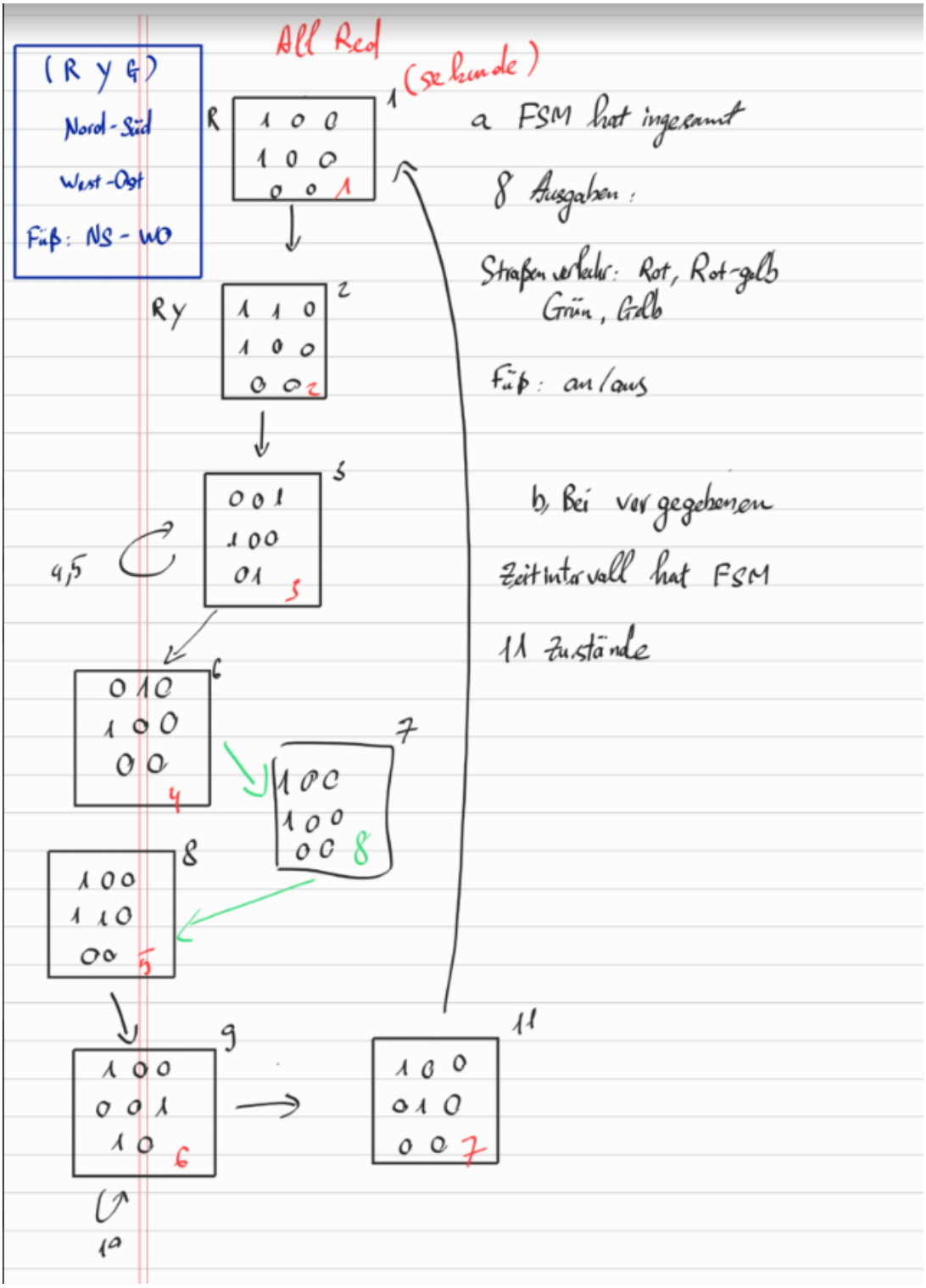
MPS Praktikum 3

Trung Thieu Quang – 771043

Viet Anh Müller - 1111510

1. Es soll eine Ampelsteuerung [1] realisiert werden, bei der in Nord-Süd-Richtung und OstWest-Richtung der Verkehr abwechselnd fließen soll:



2. Planen Sie die effiziente Implementierung! Spart viel Code und mögliche Fehler!
    a. Alle wird in Hal.c und Hal.h geschrieben

```
/* Macros for I/O */ //"u" is unsigned int
#define LOW  (0u)  //!< on (active low)
#define HIGH (1u)  //!< off (active low)

#define NORTH_SOUTH(r,y,g) switch_north_south_state(r,y,g)
#define N_R(x) Pin_N_R_Write(x)
#define N_Y(x) Pin_N_Y_Write(x)
#define N_G(x) Pin_N_G_Write(x)

#define S_R(x) Pin_S_R_Write(x)
#define S_Y(x) Pin_S_Y_Write(x)
#define S_G(x) Pin_S_G_Write(x)
// ============================================================
#define WEST(r,y,g) switch_west_state(r,y,g)
#define W_R(x) Pin_W_R_Write(x)
#define W_Y(x) Pin_W_Y_Write(x)
#define W_G(x) Pin_W_G_Write(x)


#define E_CW(x) Pin_E_CW_Write(x)
#define S_EW(x) Pin_S_CW_Write(x)

#define F_OW(x) Pin_E_CW_Write(x)
#define F_NS(x) Pin_S_CW_Write(x)
```

```
//============================================================
// State 0
#define _SET_ALL_LEDS_TO_RED ( NORTH_SOUTH(LOW, HIGH, HIGH), WEST(LOW, HIGH, HIGH), F_NS(LOW), F_OW(LOW) )
// State 1
#define _SET_NORTH_SOUTH_TO_RED_YELLOW ( NORTH_SOUTH(LOW, LOW, HIGH), WEST(LOW, HIGH, HIGH), F_NS(LOW), F_OW(LOW) )
// State 2 (North - south = grün => Fuß WO leuchten)
#define _ALLOW_NORTH_SOUTH_TO_DRIVE ( NORTH_SOUTH(HIGH, HIGH, LOW), WEST(LOW, HIGH, HIGH), F_NS(LOW), F_OW(HIGH) )
// State 3
#define _SET_NORTH_SOUTH_TO_YELLOW ( NORTH_SOUTH(HIGH, LOW, HIGH), WEST(LOW, HIGH, HIGH), F_NS(LOW), F_OW(LOW) )
// State 4
#define _SET_WEST_TO_RED_YELLOW ( NORTH_SOUTH(LOW, HIGH, HIGH), WEST(LOW, LOW, HIGH), F_NS(LOW), F_OW(LOW) )
// State 5 (OST - WEST = grün => Fuß NS leuchten)
#define _ALLOW_WEST_TO_DRIVE ( NORTH_SOUTH(LOW, HIGH, HIGH), WEST(HIGH, HIGH, LOW), F_NS(HIGH), F_OW(LOW) )
// State 6
#define _SET_WEST_TO_YELLOW ( NORTH_SOUTH(LOW, HIGH, HIGH), WEST(HIGH, LOW, HIGH), F_NS(LOW), F_OW(LOW) )
```

    b. Nutzen Sie für den 1-Sekunden Zeit-Trigger die ISR aus Termin 2

```
/**
 * Application clock interrupt service routine for isr_Clk
 *
 * @see fClock
 */
static uint32_t timer = 0;
static uint32_t zustand  = 0;
CY_ISR( IsrAppClk ) {
    count_ms++;                     // increment ms timestamp
    if ( (count_ms > 1000) ) { // next 1 s reached, set back to > 1000 as prak 2
       fsClock = 1;                 // set flag
        timer++;
        count_ms = 0;
    }
}
```

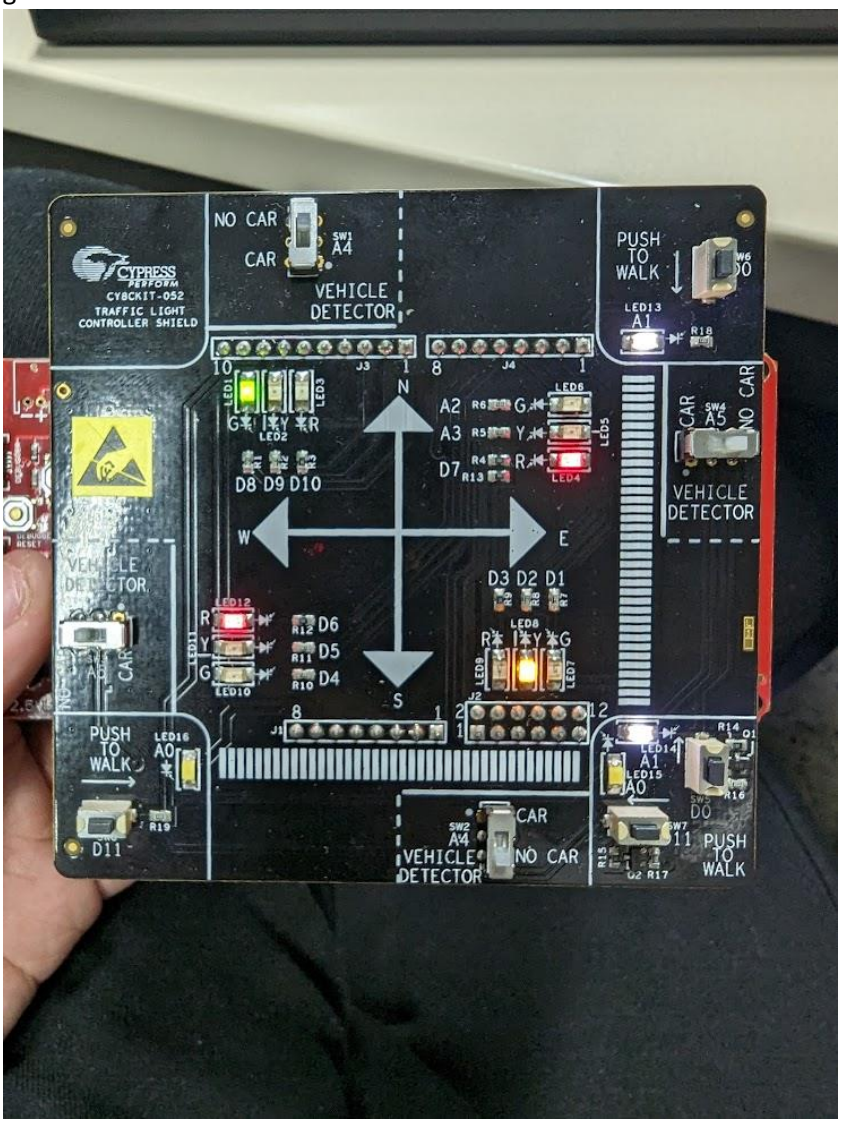
    c. Planen Sie die FSM sorgfältig:
        i. Wie können der Zustand und die Übergänge im Programm dargestellt werden? (Wieviel gibt es?): Es gibt ingesamt 8 Zustands, die nicht von Zeit abweichen. Die Übergänge können wir in der Schleife schaffen
        ii. Die Ausgaben können durch Macros und Funktionen geschafft werden

```
  // Alle rot
void cleanState(){
    _SET_ALL_LEDS_TO_RED;
}
void prepareNorthSouthToDrive(){
    _SET_NORTH_SOUTH_TO_RED_YELLOW;
}
void allowNorthSouthToDrive(){
    _ALLOW_NORTH_SOUTH_TO_DRIVE;
}
void prepareNorthSouthToStop(){
    _SET_NORTH_SOUTH_TO_YELLOW;
}

void prepareEastWestToDrive(){
    _SET_WEST_TO_RED_YELLOW;
}
void allowEastWestToDrive(){
    _ALLOW_WEST_TO_DRIVE;
}
void prepareEastWestToStop(){
    _SET_WEST_TO_YELLOW;
}
```

3. Starten Sie PSoC-Creator und laden Sie das Projekt Termin 3: MPS_W22_Prakt_3.

   c. In diesem Praktikum verwirre ich mich ein bischen, da ich nicht weiß, ob ich den Verkehr nach Deutschland oder UK simuliert soll, hier habe ich allerdings nach Deutschland gebildet



4. PWM: betrachten Sie TopDesign.cysch.
   a. Periode muss auf 999 eingestellt werden, um eine Wiederholung von etwa 10ms zu erreichen
   b. Hier schätze ich, dass mit dem Wert zwischen 0-800 wir die Helligkeit der gelben LED (east) verändern können

```c
//somehow I can't set brightness = PWM_ReadCompare() here
static uint16 brightness = 300;
// change the avarage power to change brightness
void brightness_settings(u_int input){
    brightness = PWM_ReadCompare();
    if(input && brightness < 1000){
        //uint16 test = PWM_ReadCompare();
        brightness = brightness + 200;
        PWM_WriteCompare( brightness);
        return;

    }else if(!input && brightness <= 100){
        brightness = brightness + 100;
        PWM_WriteCompare(brightness);
        return;
    }
    else if(!input) {
        brightness = brightness -200;
        PWM_WriteCompare(brightness);
        return;
    }
}
```

c. Verändern Sie die Helligkeit der LED über die Menusteuerung, z.B. ‘+‘ und ‘-‘

```c
    case '+':
        sprintf(buffer, "brightness: %d \n", brightness);
        UART_PutString(buffer);
        brightness_settings(0);
        break;
    case '-':
        sprintf(buffer, "brightness: %d \n", brightness);
        UART_PutString(buffer);
        brightness_settings(1);
        break;

        // ... und so weiter ...
    default:
        UART_PutChar( c );    // Buchstabe auf Bildschirm ausgeben
        break;
} // end switch
```

5. Erweiterungen
    a. …
    b. Fußgängeranforderung Pin_E_CW über ISR aus Termin 2: Bei Drücken des Buttons (D11 auf dem Board) sollen die Fußgänger schneller weiß bekommen oder länger weiß erhalten.

```c
if(fCWEW_Isr){
    // fast foward to turn on led Fuß ost-west
    if (zustand == 0) { // alle rot -> direkt NORD-SÜD auf gelb -> state 2 wird aktiviert
        prepareNorthSouthToDrive();
        zustand = 2;
        timer = 0;
    }
    // led Fuß ost-west lasts longer
    if (zustand == 3 && timer < 3) {
        timer = timer - 2;
    }
    // fast foward for led fuß nord - süd
    if (zustand == 4) { // alle rot -> direkt OST-WEST auf gelb -> state 6 wird aktiviert
        prepareEastWestToDrive();
        zustand = 6;
        timer = 0;
    }
    // led Fuß nord - süd lasts longer
    if (zustand == 7 && timer < 2) {
        timer = timer - 2;
    }
    fCWEW_Isr = 0; // Wieder auf Null!
}
```

    c. Die rote LED an einem zweiten PWM-Kanal anschließen