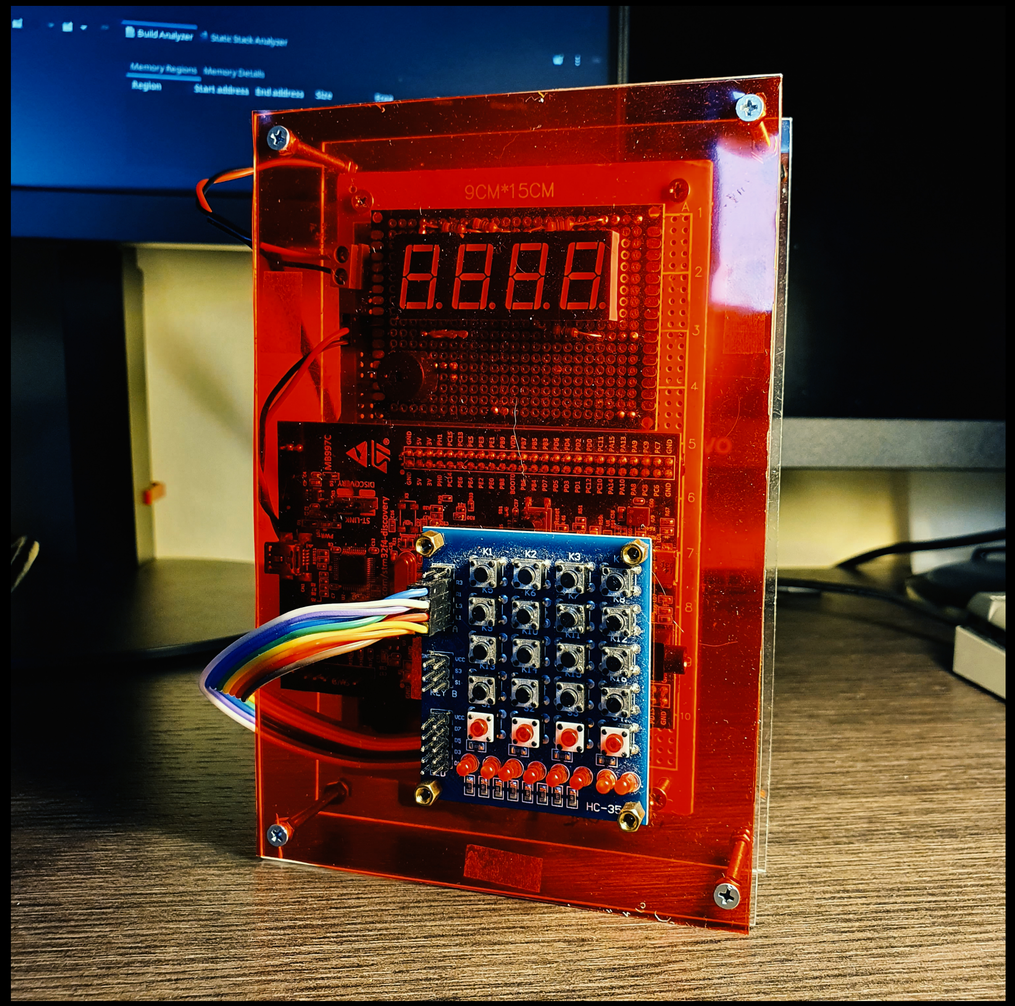
A Programmable timer for the darkroom

PROS

Università degli studi di Milano

Gabriele Esposito 15/11/21

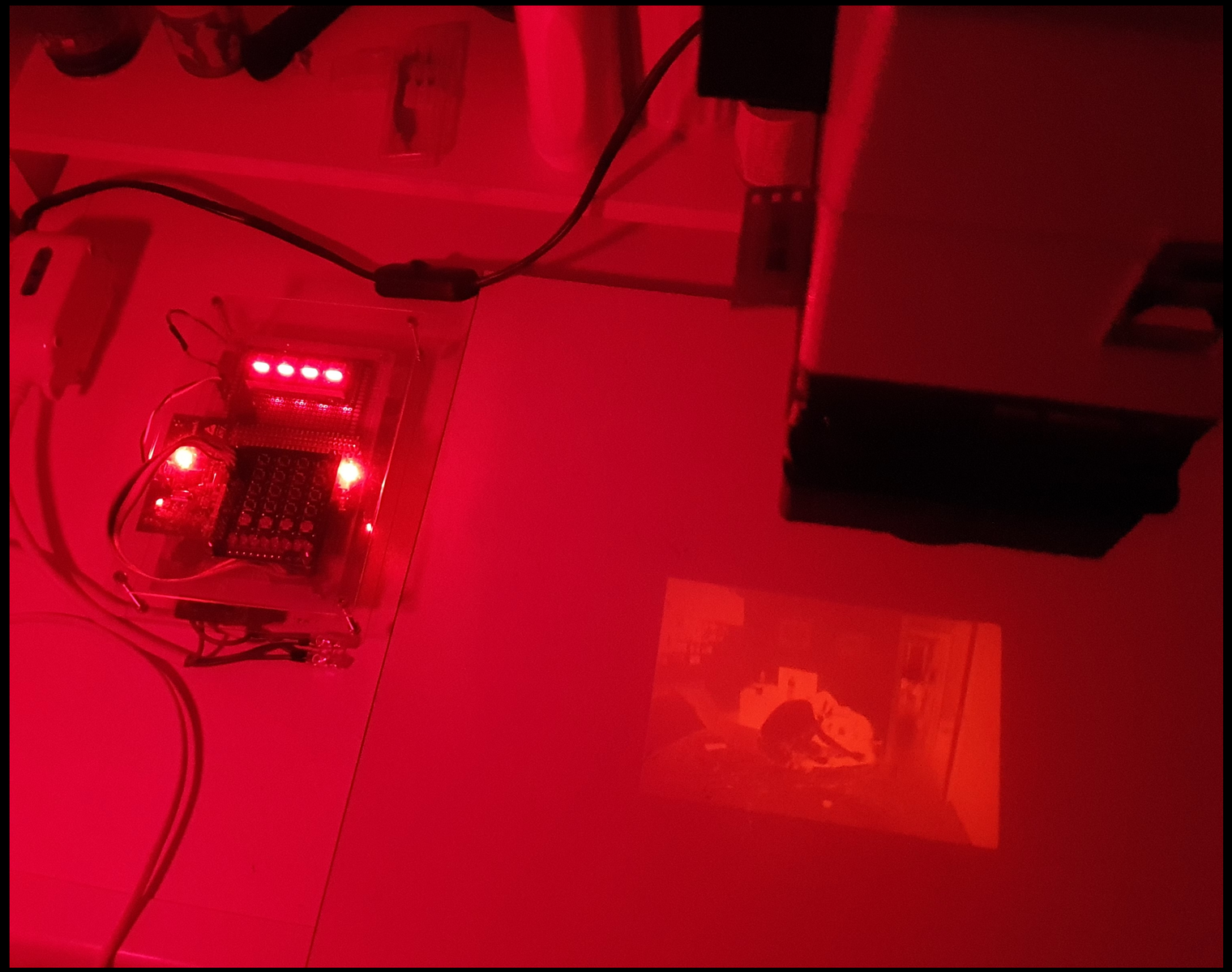
Objective

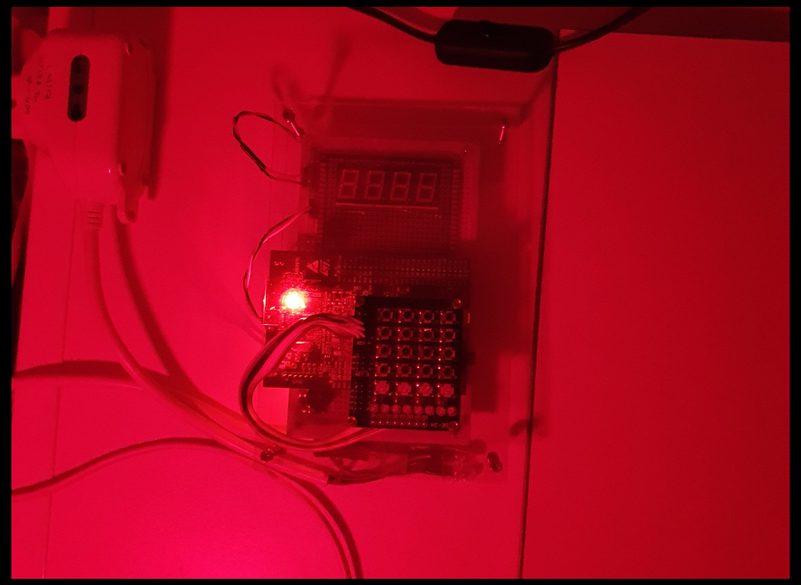

9CM*15CM
K1
K2
K3

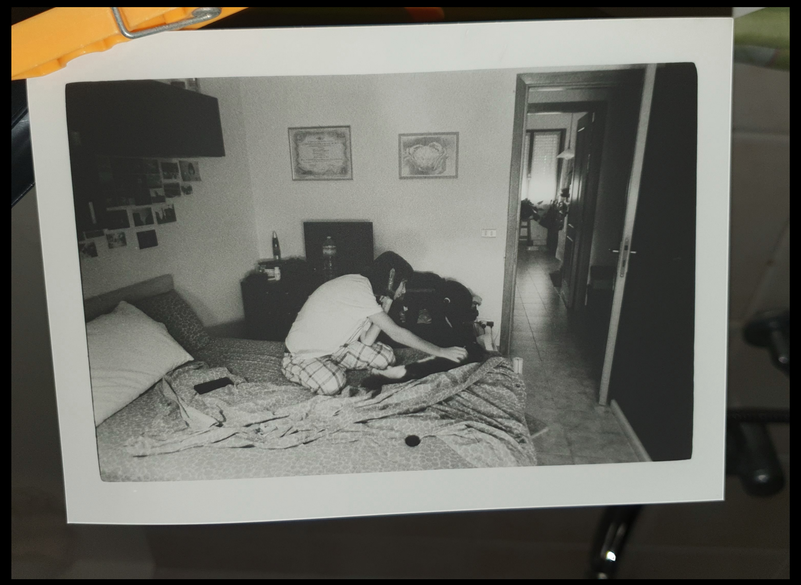

KAISER digital timer
time
reset
start
stop


ENLARGER
SAFELIGHT
FUSE MAX. 5 AMP.
IC ENLARGER TIMER
POWER
SEC.
SEC.
FOCUS
TIMER
START

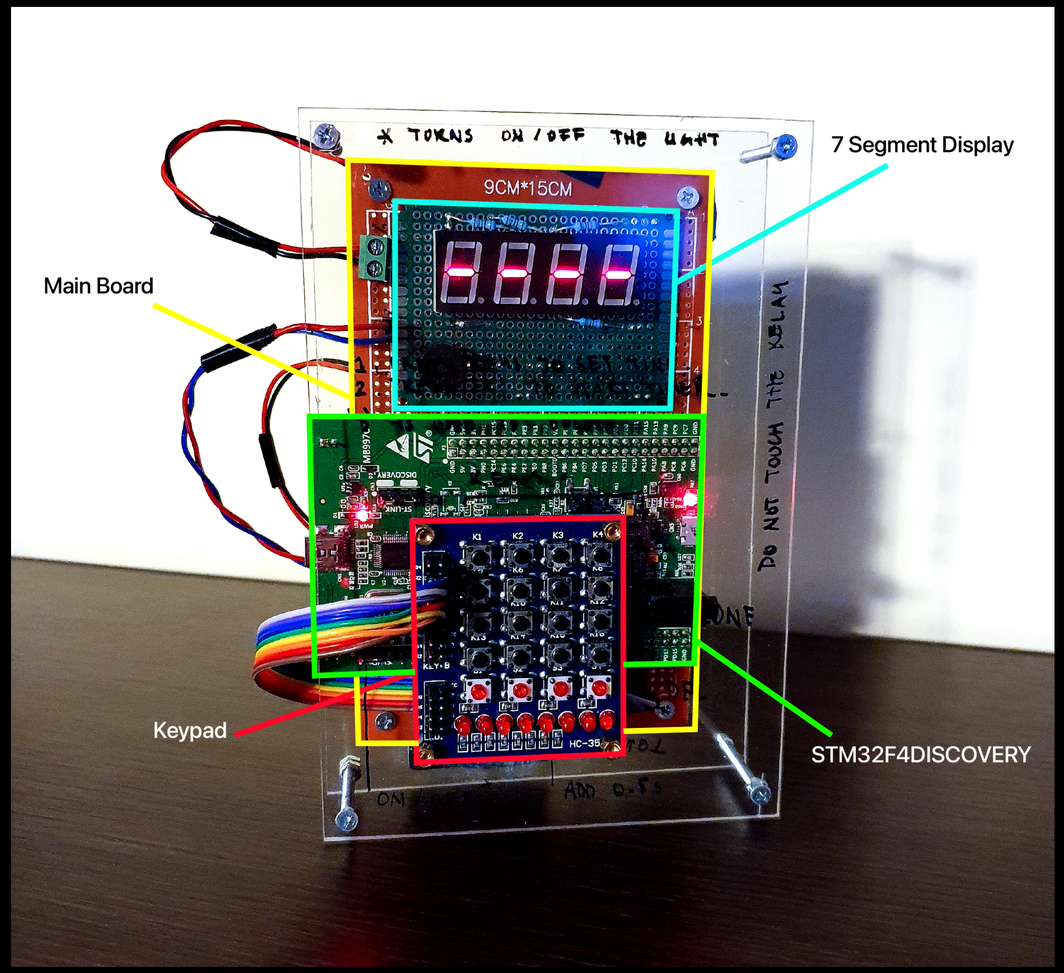

PATERSON Enlarger Timer
2000D
9.9
Brightness
Time
Reset
Focus
Run

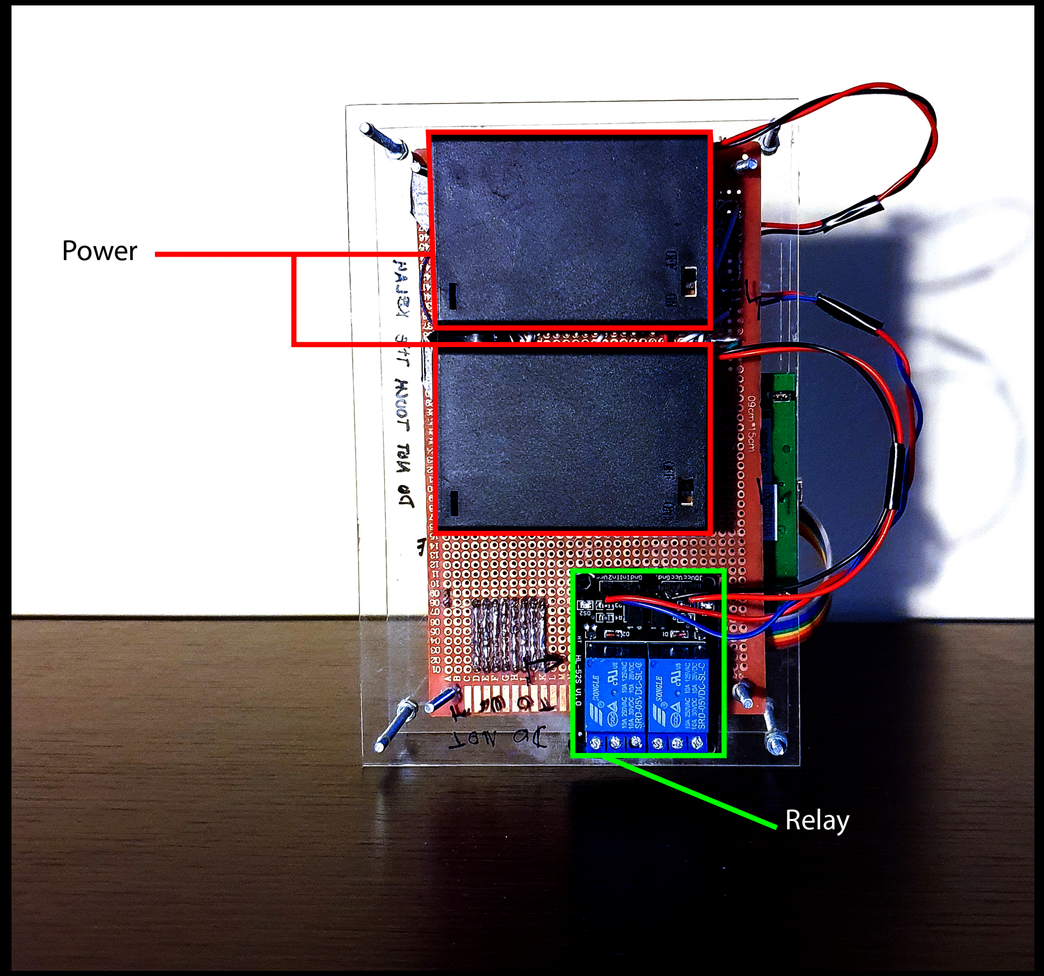




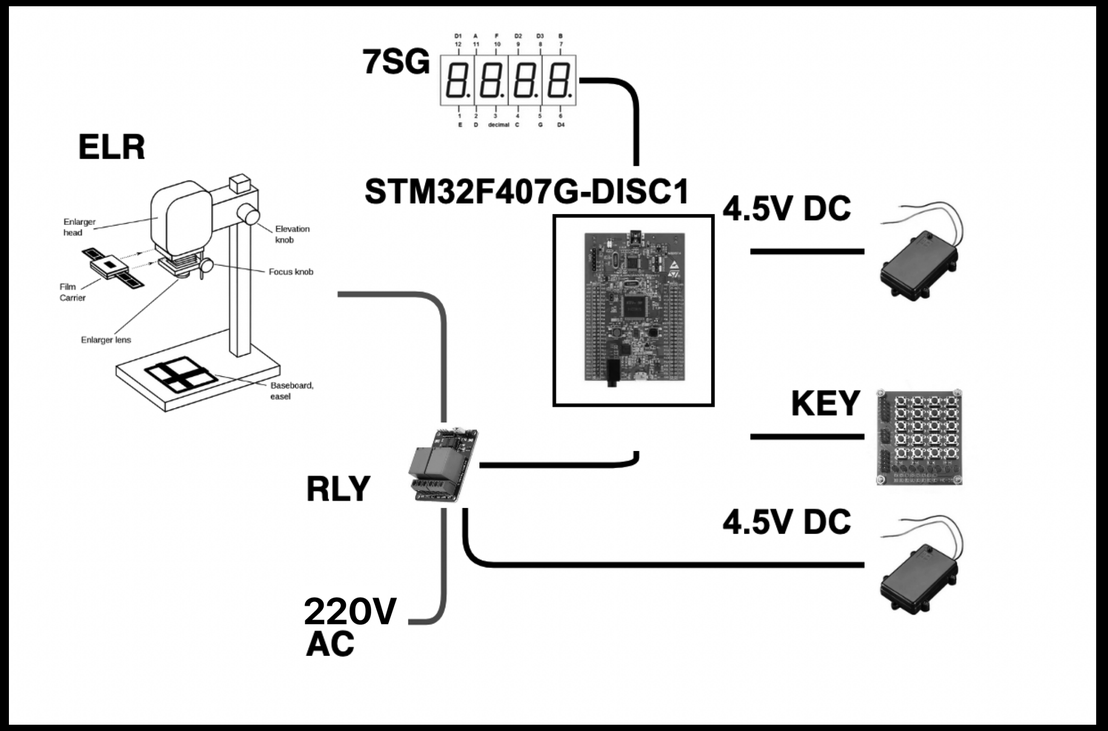


Hardware : Overview

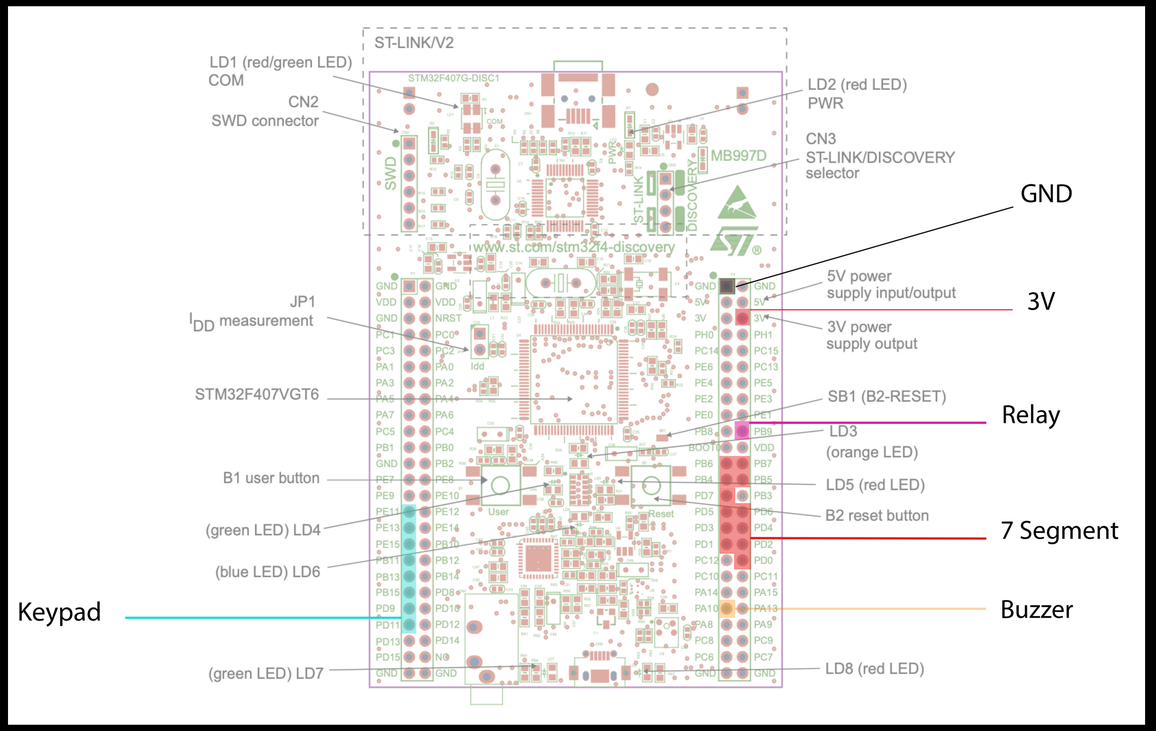




Hardware : Layout



High level component scheme



STM32F4 GPIO connections

Software

- One singleton templated class [Timer.h] containing timer logic
- Public static member functions
- 2 Basic Timers as template arguments (TIM6/TIM7)
- Drivers definitions in 7seg.h/.cpp and Keypad.h

```
//Defines what the template arguments of the timer are
typedef Timer<&htim6, &htim7> timer;
```

Timer typedef [main.cpp]

```
while (1)
{
        //Create and run the timer
        timer::run();
}
```

While loop [main.h]

```
void runImpl() {
          executeImpl();
          if (getTimeImpl() > 0) {
                    IdisplayTime();
          } else {
                    IdisplayWait();
          }
}
```

Timer.h/runImpl()

I/O Driver 7-segment

```
//write leading digits as off
for (ind = 0; ind < 4-n_digits; ++ind) {
        HAL_Delay(2);
        (*writeDigit[ind])();
        print_OFF();
}

//write number digits
for (int i = ind; i < 4; ++i) {
        HAL_Delay(2);
        (*writeDigit[i])();
        if (i == 2) {
                print_decimal();
                HAL_Delay(2);
        }
        switch (d[3-i]) {
        case 0:
                print_0();
                break;
        case 1:
                print_1();
                break;
```

Write number [Timer.h/displayTime()]

```
void write_D4()
{
        HAL_GPIO_WritePin(D_PORT, D1_PIN, GPIO_PIN_SET);
        HAL_GPIO_WritePin(D_PORT, D2_PIN, GPIO_PIN_SET);
        HAL_GPIO_WritePin(D_PORT, D3_PIN, GPIO_PIN_SET);
        HAL_GPIO_WritePin(D_PORT, D4_PIN, GPIO_PIN_RESET);
}

// Functions writing characters to the display
void print_0() // writing 0
{
  HAL_GPIO_WritePin(DIODE_PORT, DIODE_A_PIN, GPIO_PIN_SET);
  HAL_GPIO_WritePin(DIODE_PORT, DIODE_B_PIN, GPIO_PIN_SET);
  HAL_GPIO_WritePin(DIODE_PORT, DIODE_C_PIN, GPIO_PIN_SET);
  HAL_GPIO_WritePin(DIODE_PORT, DIODE_D_PIN, GPIO_PIN_SET);
  HAL_GPIO_WritePin(DIODE_PORT, DIODE_E_PIN, GPIO_PIN_SET);
  HAL_GPIO_WritePin(DIODE_PORT, DIODE_F_PIN, GPIO_PIN_SET);
  HAL_GPIO_WritePin(DIODE_PORT, DIODE_G_PIN, GPIO_PIN_RESET);
  HAL_GPIO_WritePin(DIODE_PORT, DIODE_DP_PIN, GPIO_PIN_RESET);
}
```

Select Digit / Write character (7seg.cpp)

- Select digit
- Draw Symbol
- Select next digit
- Draw Symbol
- …

I/O Driver keypad

```
if (GPIO_Pin == GPIO_PIN_11) {
        cn = 1; //fixed DETERMINED BY GPIO_PIN
        HAL_GPIO_WritePin(Rx_PORT[rn-1], Rx_PIN[rn-1], GPIO_PIN_SET);

        //Determine which row the key is in
        while(!HAL_GPIO_ReadPin(C1_PORT, C1_PIN) && rn <= 4) {
                rn++;
                HAL_GPIO_WritePin(Rx_PORT[rn-1], Rx_PIN[rn-1], GPIO_PIN_SET);
        }

        /*
         * Debounce function is called to determine if the key press is a valid press or just a bounce
         * input is accepted only if current press is a valid press, otherwise the input is ignored
         *
         * */
        for (int i = 0; i < 15; ++i) {
                if (debounce(HAL_GPIO_ReadPin(C1_PORT, C1_PIN))) {
                        valid_press = true;
                }
                HAL_Delay(DEBOUNCE_INTERVAL);
        }

        HAL_GPIO_WritePin(R1_PORT, R1_PIN, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(R2_PORT, R2_PIN, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(R3_PORT, R3_PIN, GPIO_PIN_RESET);
        HAL_GPIO_WritePin(R4_PORT, R4_PIN, GPIO_PIN_RESET);
}
else if (GPIO_Pin == GPIO_RELAY_PIN) {
        cn = 2;
```

Keypad driver [Timer.h/getIn()] called by Keypad interrupt callback

- Determine which GPIO pin triggered the interrupt (Fix column)
- Determine row looping over same column
- If signal is sufficiently stable write corresponding digit to member variable

```
if (valid_press) {
        /*
         * The key corresponding to
         * */
        key = keypad[rn][cn];
}
```

[Timer.h/getIn()]

Timer Callbacks

- TIM6 UE every 1/10 of second

  update current exposure time when exposure begins

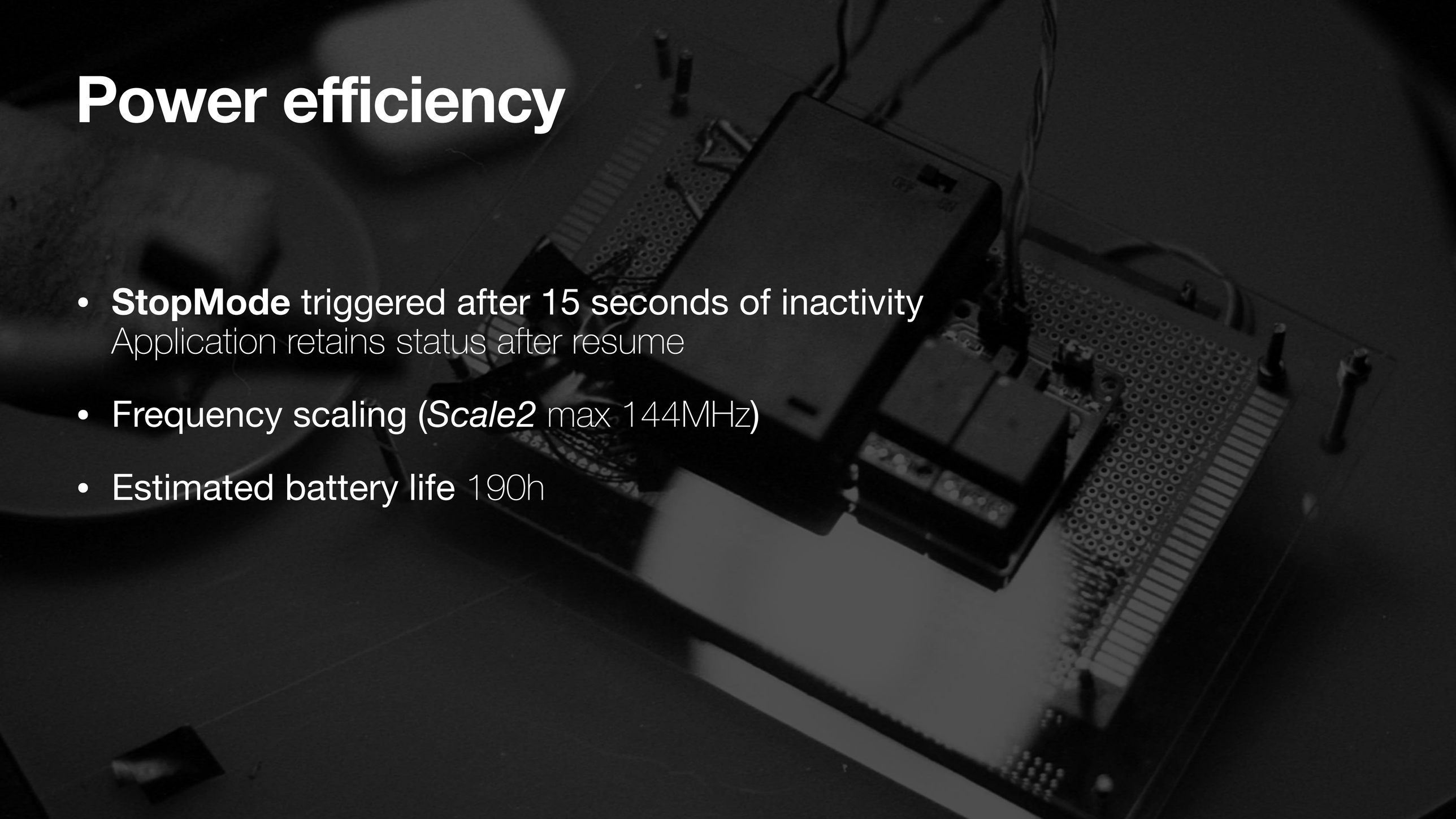
- TIM7 UE every second

  update time of inactivity if threshold is reached MCU switches to StopMode

```
void HAL_TIM_PeriodElapsedCallback(TIM_HandleTypeDef *htim) {
        if (htim == &htim7) {
                //update time of inactivity
                if(timer::inactivityTimeUpdate()) {
                        /*if inactivity time threshold is reached, MCU goes in stop mode*/
                        stopMode();
                }
        } else if (htim == &htim6) {
                //Update the remaining time when timer has started
                timer::updateTime();
        }
```

PeriodElapsedCallback [main.cpp]

Power efficiency



- **StopMode** triggered after 15 seconds of inactivity
  Application retains status after resume
- Frequency scaling (*Scale2* max 144MHz)
- Estimated battery life 190h

Features

Video demonstration

Darkroom programmable timer