Домашняя работа к занятию №3 по курсу: “ Микроконтроллеры. Вводный курс”
Студент: Андрей Федоров

Задание:

Слушателям необходимо создать проект в среде разработки STM32CubeIDE с имеющейся в наличии платой Nucleo. К плате подключить семи сегментный индикатор и написать программу «Электронный кубик». В качестве кнопки для остановки счета использовать кнопку на макетной плате или внешнюю (на усмотрением слушателя).

По стечению обстоятельств, я недавно сделал устройство для тестирования адресов i2c датчиков. И так как в нем используется динамическая индикация, я решил использовать его в качестве отладочной платы для текущего ДЗ.
Устройство имеет два индикатора, тк в ДЗ не обходимо эмитировать кубик, я буду выводить две цифры от 1 до 7 и разделю их точкой (как будто мы бросили 2 кубика).
Индикация построена на сдвиговом регистре, динамическую индикацию я осуществляю двумя выводами контроллера через транзисторы.
Схема устройства прилагается.

Вывод цифр: создаю массив из шести элементов, каждый из которых содержит отображение одной цифры (1-7). Сегменты индикаторов подключены к выводам сдвигового регистра в том порядке, который позволял наиболее удобно развести плату в домашних условиях. Так как за индикацию будет отвечать прерывание то массив задаю сразу в файле stm32f0xx_it.h:

```
/* USER CODE BEGIN PV */
extern SPI_HandleTypeDef hspi1;
extern uint8_t count;
uint8_t phase = 0;
uint8_t hc595_array[6] = {
        0B10000010,
        0B11101100,
        0B11100110,
        0B10110010,
        0B01110110,
        0B01111110,
        };

/* USER CODE END PV */
```

Переменная count это то, что мы будем выводить, phase - флаг выбора текущего индикатора, ну и естественно ссылка на SPI интерфейс.

Пишем вывод в прерывании:

```
void TIM17_IRQHandler(void)
{
  /* USER CODE BEGIN TIM17_IRQn 0 */

  /* USER CODE END TIM17_IRQn 0 */
  HAL_TIM_IRQHandler(&htim17);
  /* USER CODE BEGIN TIM17_IRQn 1 */
    uint8_t buff[2];  // опять гуляю на широкую ногу и формирую сразу обе цифры
    buff[0] = hc595_array[count%6];  // шестеричная система исчисления от 1 до 7 :)
    buff[1] = hc595_array[count/6]+1;  // добавляю точку (нулевой бит)
    HAL_GPIO_WritePin(D2_GPIO_Port, D2_Pin, 0);  // гасим оба индикатора
    HAL_GPIO_WritePin(D1_GPIO_Port, D1_Pin, 0);  // или на видео будет видна тень
соседней цифры
    HAL_GPIO_WritePin(CS_GPIO_Port, CS_Pin, 0);  // предупреждаем о том что сейчас
будем пулять данные
```

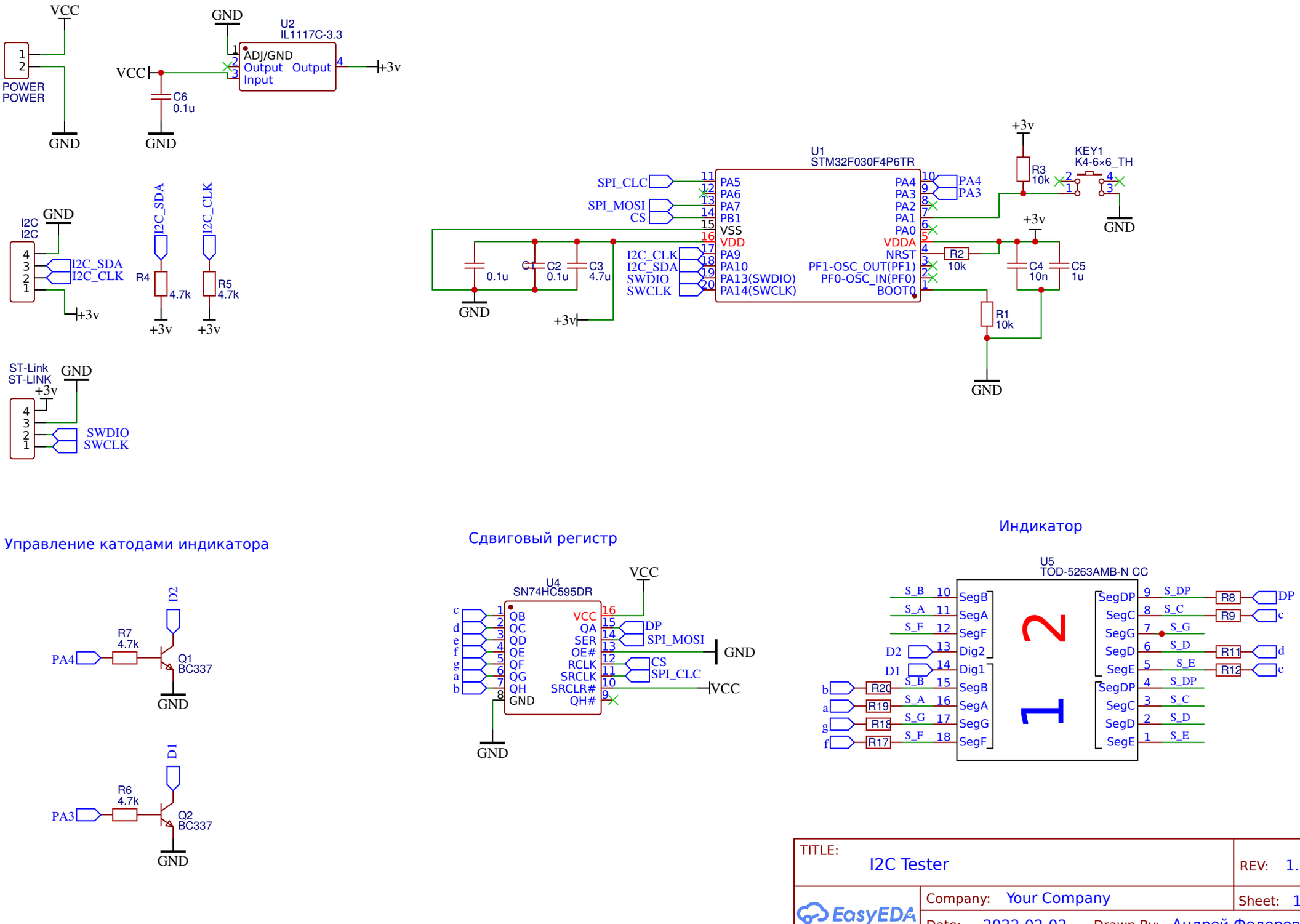
```
    HAL_SPI_Transmit(&hspi1, &buff[phase], 1, HAL_MAX_DELAY); // пуляем, при двух
индикаторах значение флага совпадает с номером элемента
    HAL_GPIO_WritePin(CS_GPIO_Port, CS_Pin, 1);  // заканчиваем нашу передачу
    HAL_GPIO_WritePin(D2_GPIO_Port, D2_Pin, phase); // зажигаем нужный индикатор
    HAL_GPIO_WritePin(D1_GPIO_Port, D1_Pin, !phase);

    phase = !phase;  // переключаем флаг

  /* USER CODE END TIM17_IRQn 1 */
}
```

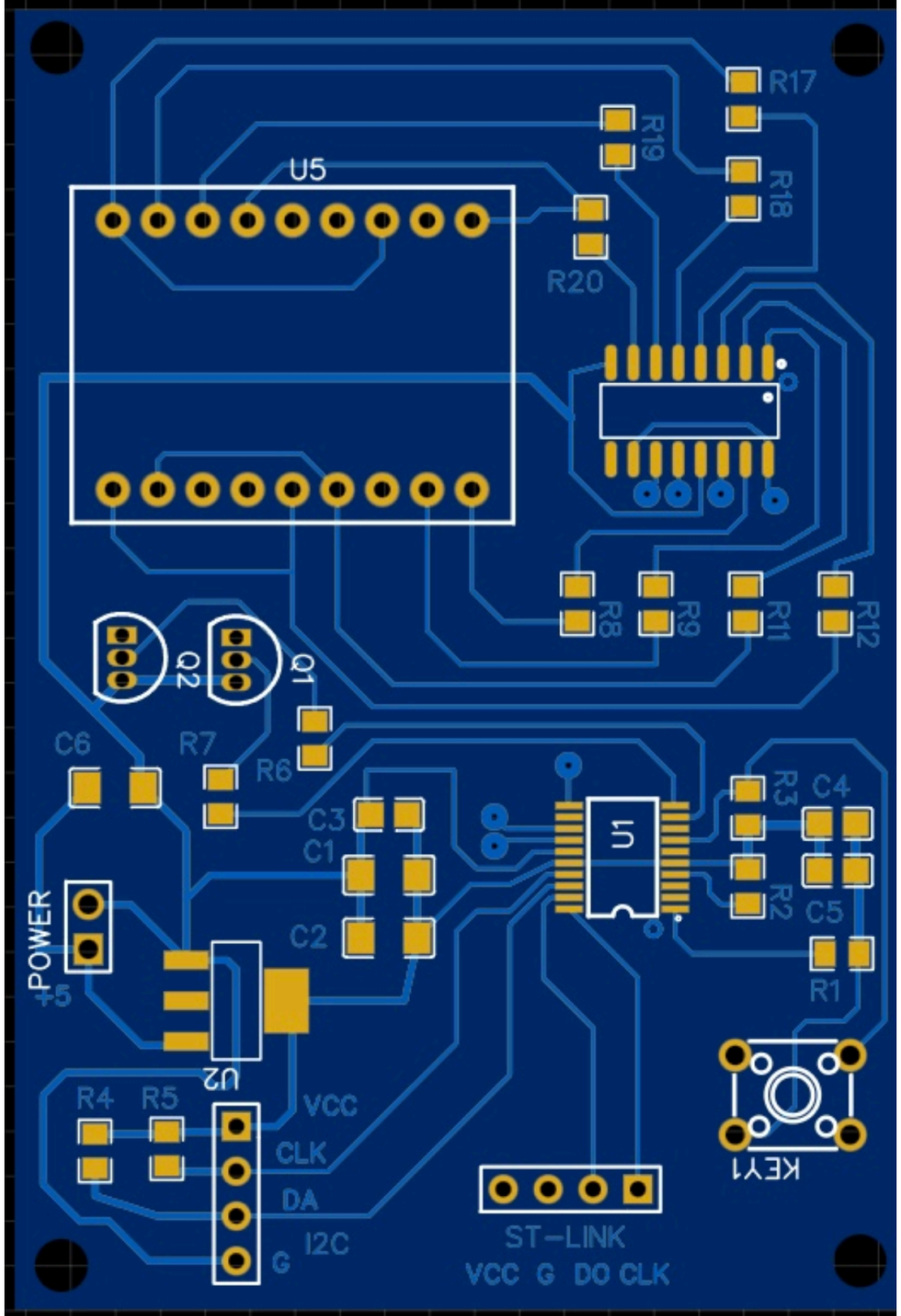
Будем показывать цифры по очереди, сначала с ускорением, а потом с замедлением, время быстрой фазы будет выбираться случайным образом, и на выходе получится два случайных числа.
Изначально этот визуальный эффект был подсмотрен в курсовой работе нашего одногрупника Петра Новикова.
main.c:

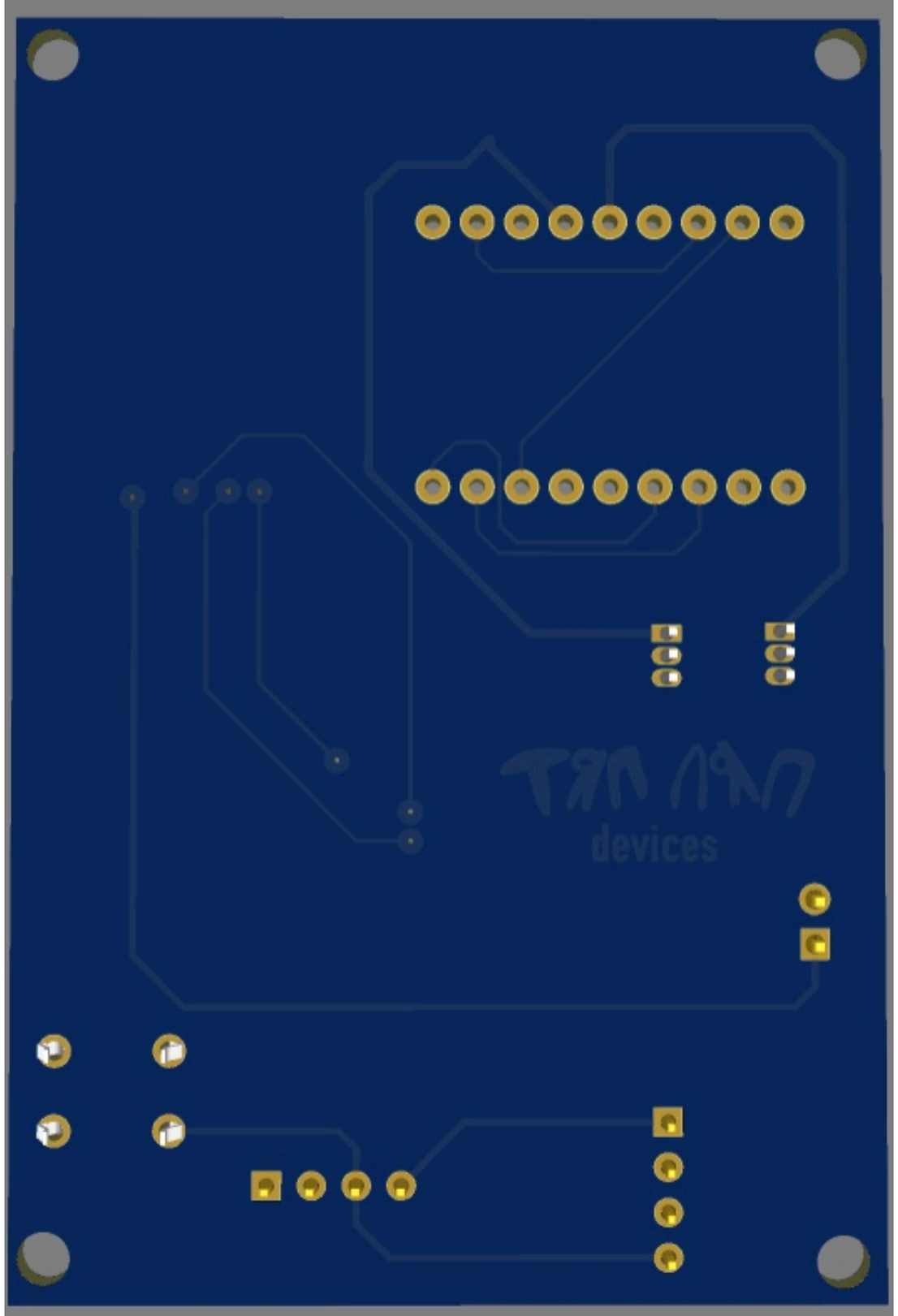
```
/* USER CODE BEGIN PV */
uint8_t count=0;
/* USER CODE END PV */
```

```
/* USER CODE BEGIN 2 */
HAL_TIM_Base_Start_IT(&htim17);  // запускаем прерывания
uint32_t next_step_1=0;
uint16_t current_delay = 200, random_delay;
uint8_t  mode =0;
/* USER CODE END 2 */
```

```
/* USER CODE BEGIN WHILE */
while (1)
{

if (HAL_GetTick()>next_step_1){
 if (mode){
       next_step_1=HAL_GetTick()+current_delay;
       count++;
       if (count>35) count = 1;
 } else if (!HAL_GPIO_ReadPin(BUTTON_GPIO_Port, BUTTON_Pin)) mode=1;

       switch (mode) {
           case 1:
               if (current_delay>10) current_delay*=.95;
               else {
                   mode=2;
                   random_delay=HAL_GetTick();
                   while (random_delay>3000)
                       random_delay=random_delay/3000+random_delay%3000;
                   random_delay+=HAL_GetTick();

               }
               break;
           case 2:
               if (HAL_GetTick()>random_delay && HAL_GPIO_ReadPin(BUTTON_GPIO_Port,
BUTTON_Pin)) mode=3;
               break;
           case 3:
               if (current_delay<200) current_delay+=8;
               else mode=0;
               break;
       }

   }

  /* USER CODE END WHILE */
```


VCC
GND
POWER
POWER
U2
IL1117C-3.3
ADJ/GND
Output
Output
Input
+3v
C6
0.1u
I2C
I2C
I2C_SDA
I2C_CLK
R4
4.7k
R5
4.7k
U1
STM32F030F4P6TR
SPI_CLC
SPI_MOSI
CS
I2C_CLK
I2C_SDA
SWDIO
SWCLK
PA5
PA6
PA7
PB1
VSS
VDD
PA9
PA10
PA13(SWDIO)
PA14(SWCLK)
PA4
PA3
PA2
PA1
PA0
VDDA
NRST
PF1-OSC_OUT(PF1)
PF0-OSC_IN(PF0)
BOOT0
R3
10k
KEY1
K4-6×6_TH
R2
10k
C1
0.1u
C2
0.1u
C3
4.7u
C4
10n
C5
1u
R1
10k
ST-Link
ST-LINK
SWDIO
SWCLK
Управление катодами индикатора
D2
R7
4.7k
PA4
Q1
BC337
D1
R6
4.7k
PA3
Q2
BC337
Сдвиговый регистр
U4
SN74HC595DR
QB
QC
QD
QE
QF
QG
QH
GND
VCC
QA
SER
OE#
RCLK
SRCLK
SRCLR#
QH#
DP
SPI_MOSI
CS
SPI_CLC
Индикатор
U5
TOD-5263AMB-N CC
SegB
SegA
SegF
Dig2
Dig1
SegB
SegA
SegG
SegF
SegDP
SegC
SegG
SegD
SegE
SegDP
SegC
SegD
SegE
R20
R19
R18
R17
R8
R9
R11
R12
DP
c
d
e
TITLE:
I2C Tester
REV: 1.0
EasyEDA
Company: Your Company
Sheet: 1/1
Date: 2022-02-02
Drawn By: Андрей Федоров


R17
R19
R18
U5
R20
R8
R9
R11
R12
Q2
Q1
C6
R7
R6
C3
C1
U1
R3
C4
R2
C5
C2
R1
POWER
+5
U2
R4
R5
VCC
CLK
DA
I2C
G
KEY1
ST-LINK
VCC G DO CLK


devices