Институт за математику и информатику

Природно-математички факултет

Универзитет у Крагујевцу

Завршни пројекат из предмета Микропроцесорски системи

Тема: Систем за управљање роботском руком помоћу степ или серво мотора

Студент:

Васиљевић Лука 51/2018

Професор:

др Александар Пеулић

Фебруар 2022.

Крагујевац

Садржај

Увод 3
Објашњење алгоритма и кода 4
Електро шема у програму Proteus 8
Упутство за коришћење система 10

Увод

Као што сам Project Charter говори, циљ овог пројекта је реализација система за управљање роботском руком за више намена. Као примере коришћења бих навео роботска рука која при убацивању жетона у апарат остварује шансу да освоји плишану играчку, односно роботска рука која се користи при лемљењу и уопштено прецизном раду са штампаним плоча у електронској индустрији (претпостављам да се концептуално мој пројекат може узети у обзир за такав подухват, а подразумева се да би се прецизност и АД конверзије и свих компонената додатно размотрила). Уз ову документацију, прилажем и Project Charter (pc.pdf и project charter.pptx), Work Breakdown Structure (WorkBreakdownStructureRoboticHand.pdf) , Product Backlog (pbl.pdf) и гантограм са током рада(.gan и .pdf формат), а пре свега тога, електро шему направљену у Proteus-u и код генерисан помоћу Stm32CubeIDE софтвера. Коришћени микропроцесор је из породице STM32, а модел STM32F103C6, док о компонентама електро-шеме ће бити више речи у наставку документације.

Објашњење алгоритма и кода

Идеја и алгоритам који стоје ван техничке реализације је следећа- пројектовани систем садржи џојстик (који се састоји од два потенциометра и једног дугмета) и одређене лед диоде и седмосегментне лед дисплеје за праћење стања. Коришћењем џојстика управља се роботском руком, а притиском дугмета роботска рука се спушта на подлогу, захвата и враћа у почетни положај.

Што се самог кода тиче, урађено је следеће:

- Иницијализација седмосегментних лед дисплеја

```
319 void initAxes(void) {
320     HAL_GPIO_WritePin(DISPLAY_PORT, XJ1, GPIO_PIN_SET);
321     HAL_GPIO_WritePin(DISPLAY_PORT, XJ2, GPIO_PIN_SET);
322     HAL_GPIO_WritePin(DISPLAY_PORT, XJ3, GPIO_PIN_SET);
323     HAL_GPIO_WritePin(DISPLAY_PORT, XJ4, GPIO_PIN_SET);
324
325     HAL_GPIO_WritePin(DISPLAY_PORT, XD1, GPIO_PIN_SET);
326     HAL_GPIO_WritePin(DISPLAY_PORT, XD2, GPIO_PIN_SET);
327     HAL_GPIO_WritePin(DISPLAY_PORT, XD3, GPIO_PIN_SET);
328     HAL_GPIO_WritePin(DISPLAY_PORT, XD4, GPIO_PIN_SET);
329
330     HAL_GPIO_WritePin(DISPLAY_PORT, YJ1, GPIO_PIN_SET);
331     HAL_GPIO_WritePin(DISPLAY_PORT, YJ2, GPIO_PIN_SET);
332     HAL_GPIO_WritePin(DISPLAY_PORT, YJ3, GPIO_PIN_SET);
333     HAL_GPIO_WritePin(DISPLAY_PORT, YJ4, GPIO_PIN_SET);
334
335     HAL_GPIO_WritePin(DISPLAY_PORT, YD1, GPIO_PIN_SET);
336     HAL_GPIO_WritePin(DISPLAY_PORT, YD2, GPIO_PIN_SET);
337     HAL_GPIO_WritePin(DISPLAY_PORT, YD3, GPIO_PIN_SET);
338     HAL_GPIO_WritePin(DISPLAY_PORT, YD4, GPIO_PIN_SET);
339
340     HAL_Delay(300);
341
342     HAL_GPIO_WritePin(DISPLAY_PORT, XJ1, GPIO_PIN_RESET);
343     HAL_GPIO_WritePin(DISPLAY_PORT, XJ2, GPIO_PIN_RESET);
344     HAL_GPIO_WritePin(DISPLAY_PORT, XJ3, GPIO_PIN_RESET);
345     HAL_GPIO_WritePin(DISPLAY_PORT, XJ4, GPIO_PIN_RESET);
346
347     HAL_GPIO_WritePin(DISPLAY_PORT, XD1, GPIO_PIN_RESET);
348     HAL_GPIO_WritePin(DISPLAY_PORT, XD2, GPIO_PIN_RESET);
349     HAL_GPIO_WritePin(DISPLAY_PORT, XD3, GPIO_PIN_RESET);
350     HAL_GPIO_WritePin(DISPLAY_PORT, XD4, GPIO_PIN_RESET);
351
352     HAL_GPIO_WritePin(DISPLAY_PORT, YJ1, GPIO_PIN_RESET);
353     HAL_GPIO_WritePin(DISPLAY_PORT, YJ2, GPIO_PIN_RESET);
354     HAL_GPIO_WritePin(DISPLAY_PORT, YJ3, GPIO_PIN_RESET);
355     HAL_GPIO_WritePin(DISPLAY_PORT, YJ4, GPIO_PIN_RESET);
356
357     HAL_GPIO_WritePin(DISPLAY_PORT, YD1, GPIO_PIN_RESET);
358     HAL_GPIO_WritePin(DISPLAY_PORT, YD2, GPIO_PIN_RESET);
359     HAL_GPIO_WritePin(DISPLAY_PORT, YD3, GPIO_PIN_RESET);
360     HAL_GPIO_WritePin(DISPLAY_PORT, YD4, GPIO_PIN_RESET);
361
362 }
```

Слика 1 Иницијализација седмосегментних дисплејева

- С обзиром да су коришћени BCD (Binary-Coded Decimal) седмосегментни дисплеји било је неопходно извршити мапирање простог int типа у C програмском језику у бинарни и сходно томе слати сигнал за приказ одређене цифре

```
/* USER CODE BEGIN Includes */
#define CHECK_BIT(var,pos) ((var) & (1<<(pos)))

#define ADC_PORT GPIOA
#define DISPLAY_PORT GPIOB
// X osa jedinice
#define XJ1 GPIO_PIN_0
#define XJ2 GPIO_PIN_1
#define XJ3 GPIO_PIN_2
#define XJ4 GPIO_PIN_3
// X osa desetice
#define XD1 GPIO_PIN_4
#define XD2 GPIO_PIN_5
#define XD3 GPIO_PIN_6
#define XD4 GPIO_PIN_7
// Y osa jedinice
#define YJ1 GPIO_PIN_8
#define YJ2 GPIO_PIN_9
#define YJ3 GPIO_PIN_10
#define YJ4 GPIO_PIN_11
// Y osa desetice
#define YD1 GPIO_PIN_12
#define YD2 GPIO_PIN_13
#define YD3 GPIO_PIN_14
#define YD4 GPIO_PIN_15
// led za belezenje konverzije
#define ADCX GPIO_PIN_1
#define ADCY GPIO_PIN_3
// interrupt pin
#define INTERR_PIN GPIO_PIN_0
/* USER CODE END Includes */
```

Слика 2 Макро за проверавање битова и константе у систему

```
void setAxisJ(int partialCoordinate, char axis) {
    if (axis == 'X') {
        HAL_GPIO_WritePin(DISPLAY_PORT, XJ1,
        CHECK_BIT(partialCoordinate, 0) == 0 ? GPIO_PIN_RESET : GPIO_PIN_SET);
        HAL_GPIO_WritePin(DISPLAY_PORT, XJ2,
        CHECK_BIT(partialCoordinate, 1) == 0 ? GPIO_PIN_RESET : GPIO_PIN_SET);
        HAL_GPIO_WritePin(DISPLAY_PORT, XJ3,
        CHECK_BIT(partialCoordinate, 2) == 0 ? GPIO_PIN_RESET : GPIO_PIN_SET);
        HAL_GPIO_WritePin(DISPLAY_PORT, XJ4,
        CHECK_BIT(partialCoordinate, 3) == 0 ? GPIO_PIN_RESET : GPIO_PIN_SET);
    } else if (axis == 'Y') {
        HAL_GPIO_WritePin(DISPLAY_PORT, YJ1,
        CHECK_BIT(partialCoordinate, 0) == 0 ? GPIO_PIN_RESET : GPIO_PIN_SET);
        HAL_GPIO_WritePin(DISPLAY_PORT, YJ2,
        CHECK_BIT(partialCoordinate, 1) == 0 ? GPIO_PIN_RESET : GPIO_PIN_SET);
        HAL_GPIO_WritePin(DISPLAY_PORT, YJ3,
        CHECK_BIT(partialCoordinate, 2) == 0 ? GPIO_PIN_RESET : GPIO_PIN_SET);
        HAL_GPIO_WritePin(DISPLAY_PORT, YJ4,
        CHECK_BIT(partialCoordinate, 3) == 0 ? GPIO_PIN_RESET : GPIO_PIN_SET);
    }
}
```

Слика 3 Писање вредности за јединицу координата

- Урађена је аналогно-дигитална конверзија 2 канала (за x и y осу, респективно) и учитавање сирове (raw) вредности у промењиве па прилагођавање на скалу 0-99 (ограничено је због недовољно великог броја пинова)

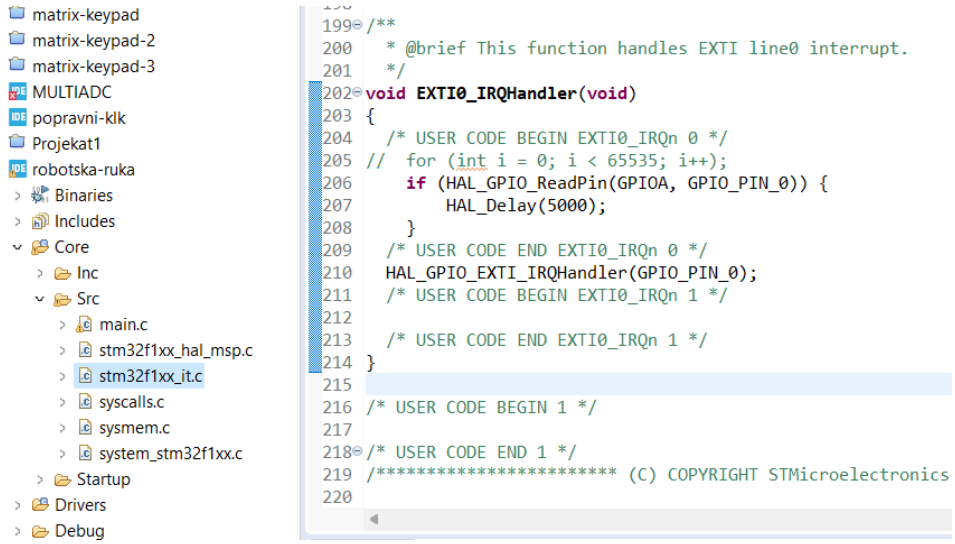
```
410 void getADCValues(uint32_t *adcResultX, uint32_t *adcResultY) {
411
412     HAL_GPIO_WritePin(ADC_PORT, ADCX, GPIO_PIN_SET);
413     HAL_GPIO_WritePin(ADC_PORT, ADCY, GPIO_PIN_SET);
414
415     HAL_Delay(300);
416     ADC_SELECT_CH0();
417     HAL_ADC_Start(&hadc1);
418     HAL_ADC_PollForConversion(&hadc1, 1000);
419     *adcResultX = HAL_ADC_GetValue(&hadc1);
420     HAL_ADC_Stop(&hadc1);
421
422     ADC_SELECT_CH1();
423     HAL_ADC_Start(&hadc1);
424     HAL_ADC_PollForConversion(&hadc1, 1000);
425     *adcResultY = HAL_ADC_GetValue(&hadc1);
426     HAL_ADC_Stop(&hadc1);
427
428     HAL_GPIO_WritePin(ADC_PORT, ADCX, GPIO_PIN_RESET);
429     HAL_GPIO_WritePin(ADC_PORT, ADCY, GPIO_PIN_RESET);
430     HAL_Delay(300);
431 }
```

Слика 4 Рачунање сирових вредности конверзије

```
156     while (1) {
157         getADCValues(&adcResultX, &adcResultY);
158
159         int xValue = (adcResultX * 100) / 4095;
160         setAxis(xValue == 100 ? 99 : xValue, 'X');
161         int yValue = (adcResultY * 100) / 4095;
162         setAxis(yValue == 100 ? 99 : yValue, 'Y');
163     /* USER CODE END WHILE */
164
165     /* USER CODE BEGIN 3 */
166     }
167   /* USER CODE END 3 */
168 }
```

Слика 5 Мапирање сирових вредности у одговарајући интервал и позивање функције за испис координата

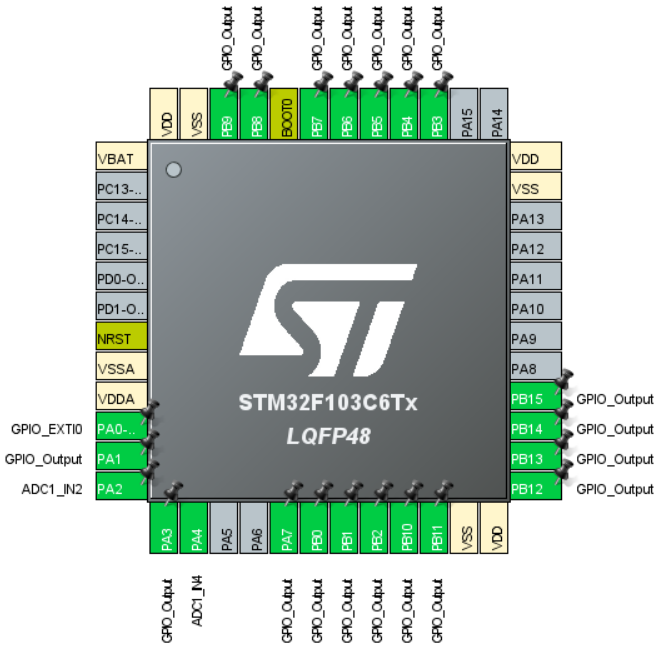
- Направљена је процедура за обраду прекида која при спуштању роботске руке зауставља мерење на одређено време (време које је неопходно роботској руци да се спусти, захвати и врати у првобитни положај)



Слика 6 Реализација прекида у коду

Везано за хардверску конфигурацију микропроцесора, важно је поменути:

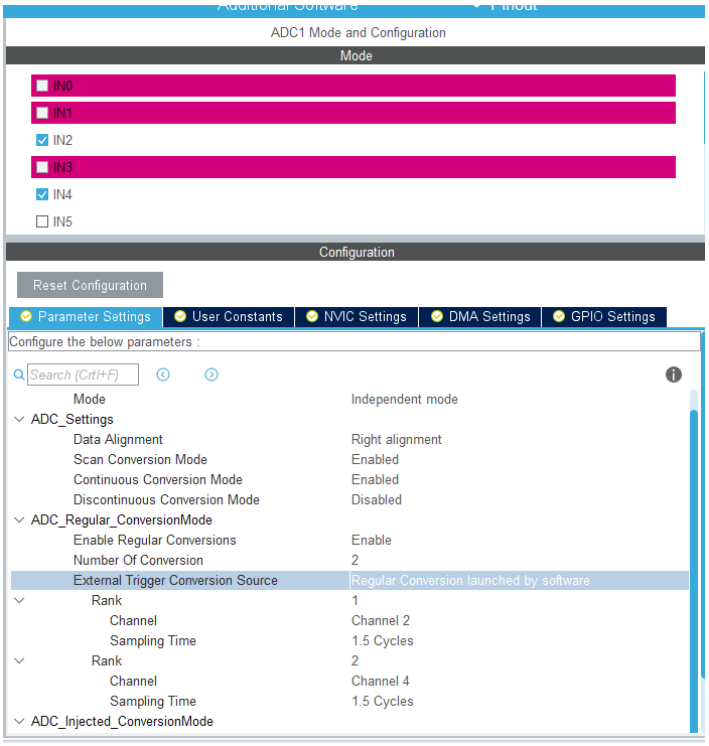
- Подразумевано додељивање пинова у потребну сврху (2 пина за аналогно-дигиталну конверзију, један за прекид, и остатак за излаз)



Слика 7 Пинови коришћеног микропроцесора

- Скривени пинови за уземљење и напон (VSSA и VDDA)
- Додатна подешавања у GPIO и NVIC одељку за обраду прекида

- Додатна подешавања у ADC1 одељку за реализацију два канала (Number of Conversions, Scan Conversion Mode, Continuous Conversion Mode)

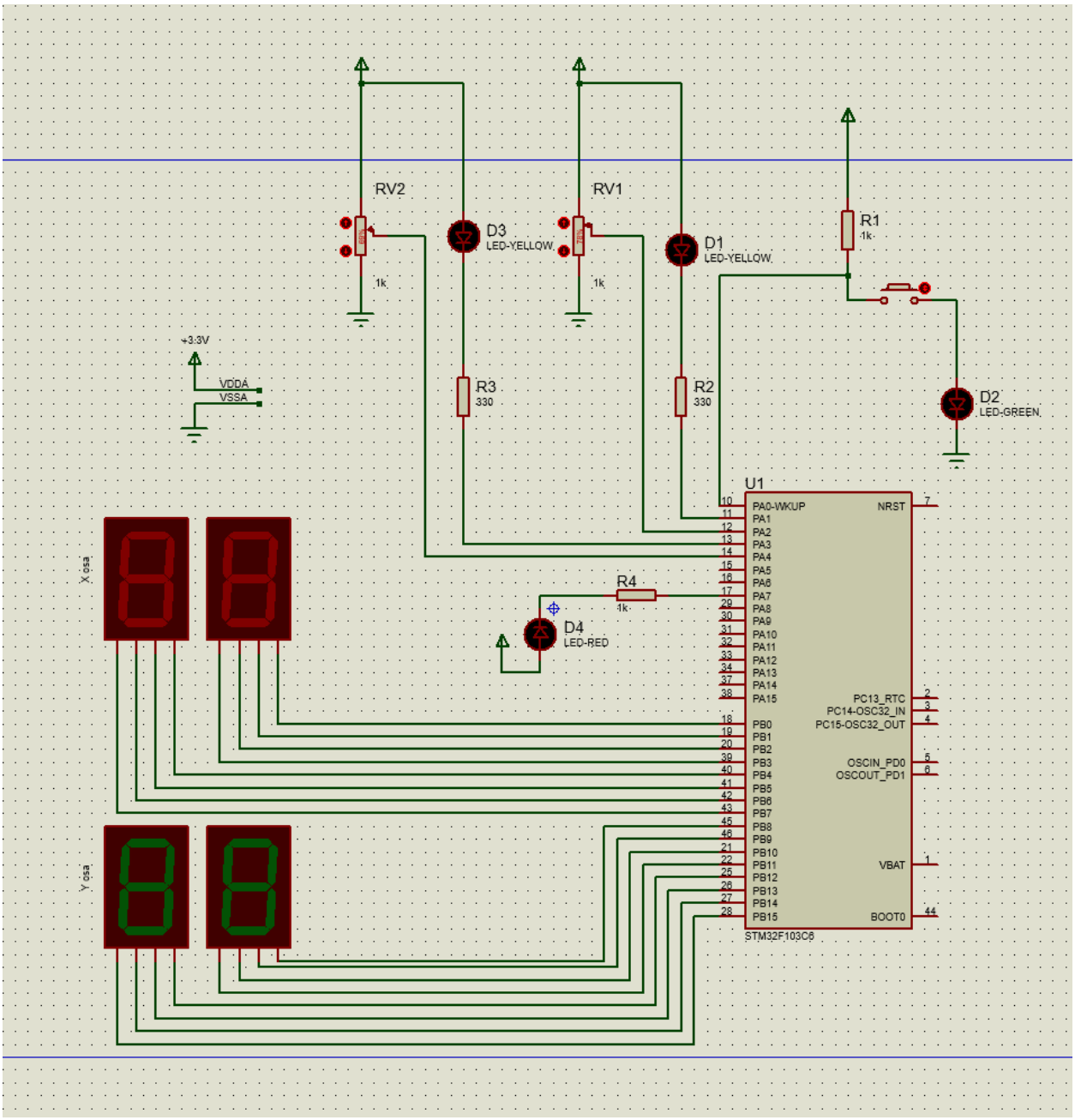


Слика 8 Конфигурација двоканалне аналогно-дигиталне конверзије

Електро шема у програму Proteus

Координате се задају преко потенциометара (координате за x и y осу, да будемо потпуно прецизни). Дугме служи за спуштање роботске руке на подлогу, а помоћу црвене лед диоде која симулира прекид ми знамо тачно у ком тренутку роботска рука почиње процес спуштања, хватања и подизања у почетни положај. Седмосегментни дисплеји служе за приказивање тренутне координате на x и y оси због немогућности хардверске реализације. Црвени седмосегментни дисплеји приказују координате x осе, док зелени приказују координате y осе. Црвено дугме представља ухваћен прекид, а зелено је притиснуто дугме за спуштање, хватање и враћање у првобитни положај роботске руке. Одвојени пинови VSSA и VDDA представљају уземљење и напон. На следећој слици се налази списак коришћених компонената.

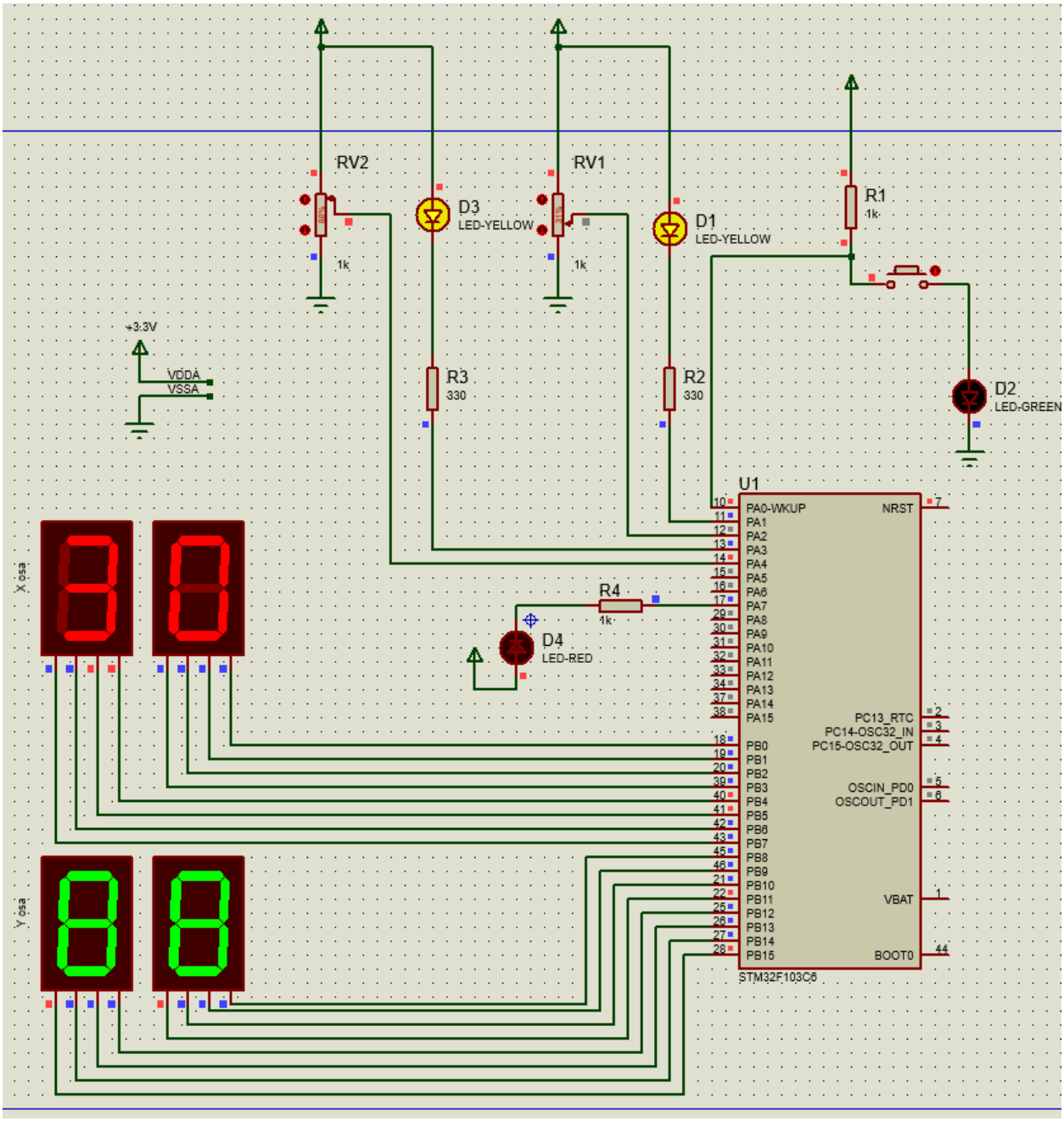
Слика 9 Списак компонената коришћених при изради система



Слика 10 Изглед шеме у програму Proteus

Упутство за коришћење система

Као што је наведено, покренута симулација у програму Proteus мења заправо хардверску реализацију. Померањем потенциометара RV-1 и RV-2 померамо роботску руку по осама. Кликом на дугме правимо прекид током ког роботска рука врши оно за шта је намењена, а онемогућава се њено померање док се, наравно, не изврши акција у прекиду. На следећој слици се налази слика симулације у раду.



Слика 11 Изглед симулације у током рада