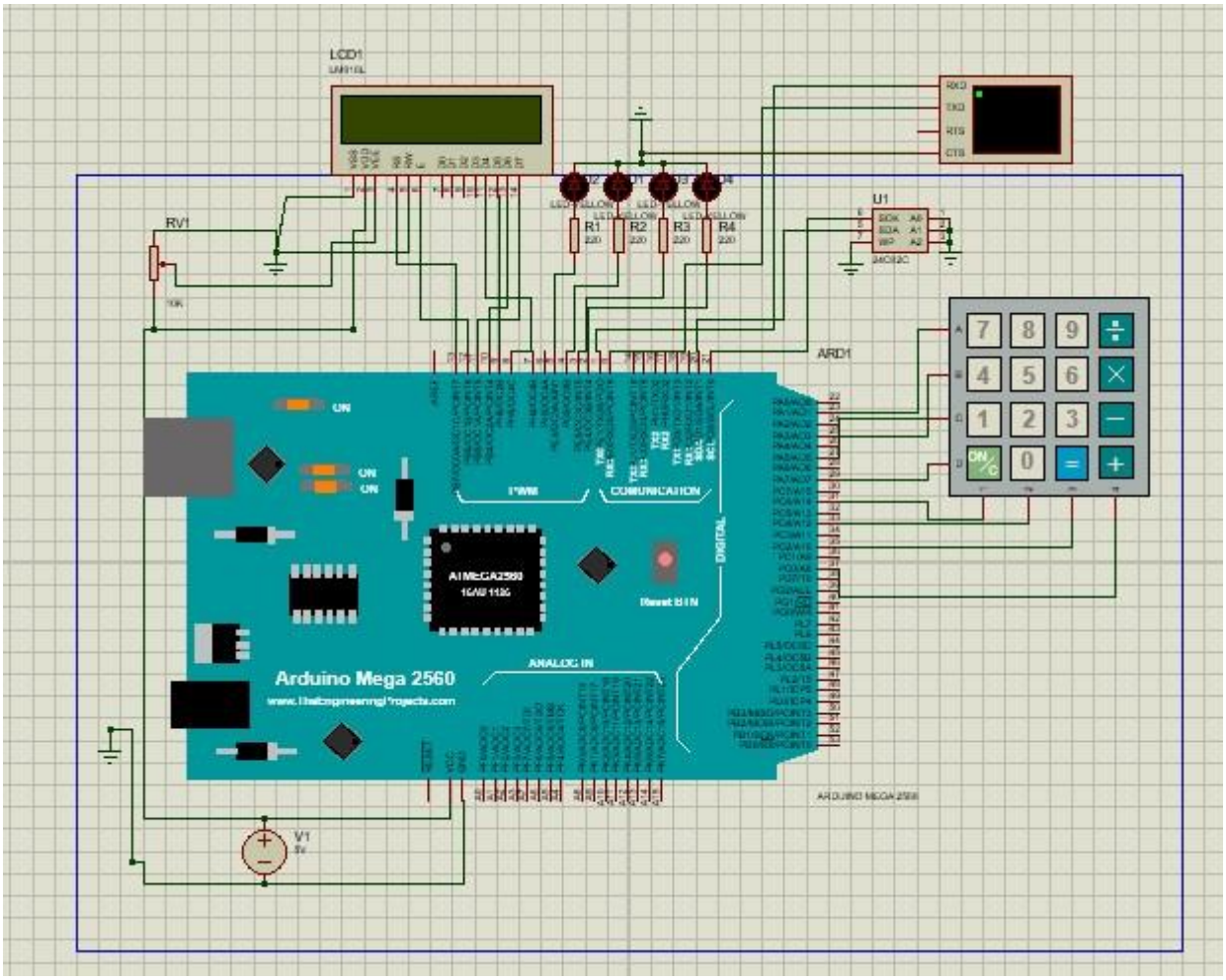
آزمایش شماره ی 7 **آزمایشگاه ریز پردازنده**

نام و نام خانوادگی: هلیا سادات هاشمی پور **شماره دانشجویی:**9831106

نام استاد:استاد علیزاده **تاریخ:**3 آذر 1400

مدار پیاده سازی شده به صورت زیر هست:



کد

ابتدا کتابخانه‌های مورد نظر (شامل کتابخانه Wire.h برای خواندن و نوشتن در حافظه) را اضافه می‌کنیم. سپس آدرس‌های مورد نظر را که مقدار ثابت دارند در ابتدا مشخص می‌کنیم و حالت‌هایی که ماشین می‌تواند در آن ها قرار بگیرد را نیز با اعداد 0 تا 4 تعیین می‌کنیم. پین‌های مربوط به LEDها را تعیین کرده و زمان‌هایی که هر حالت ماشین در ان می‌ماند را در آرایه modeTimes و نام حالت ها را در آرایه modeMassages می‌ریزیم. سپس پین‌های مربوط به LCD و کیپد را تعریف می‌کنیم. متغیر currMode برای نگهداری حالت فعلی، isWashing برای تعیین اینکه ماشین در حال فعالیت است یا خیر،remainigTime برای نگهداری زمانی که برای هر حالت باید سپری شود و startTime هم برای نگه داشتن زمان تعریف می‌شوند.

```
#include <Wire.h>
#include <Keypad.h>
#include <LiquidCrystal.h>

#define DEVICE_ADDRESS 0b1010000
#define MODE_MEMORY_ADDR 100
#define PRE_WASH 0
#define DETERGENT_WASH 1
#define WATER_WASH 2
#define DRYING 3
#define FINISH 4
int ledPins[4] = {5, 4, 3, 2};
uint8_t modeTimes[4] = {4, 4, 4, 4};//in seconds
String modeMessages[5]={"PRE","DETERGENT","WATER","DRYING","FINISH"};

//Setting LCD configs
const int rs = 13, en = 12, d4 = 8, d5 = 9, d6 = 10, d7 =11;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

//Setting Keypad configs
const byte ROWS = 4; //four rows
const byte COLS = 4; //four columns
char keys[ROWS][COLS] = {
  {'7','8','9','/'},
  {'4','5','6','*'},
  {'1','2','3','-'},
  {'C','0','=','+'}
};
byte rowPins[ROWS] = {23, 25, 27, 29}; //connect to the row pinouts of the keypad
byte colPins[COLS] = {31,33, 35,37}; //connect to the column pinouts of the keypad
Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS );
int currMode = PRE_WASH;
bool isWashing = false;
int remainingTime;
unsigned long startTime = millis();
```

در تابع setup، Serial.begin برای شروع ارتباطات سریال، Wire.begin برای عملیت حافظه و lcd.begin را برای فعال کردن LCD فراخوانی می‌شوند. پین‌های مربوط به led ها به عنوان خروجی تعریف شده و lcd نیز clear می‌شود. در نهایت loadCofigs را فراخوانی می‌کنیم.

```
void setup() {
  Serial.begin(9600);

  for(int i=0; i<FINISH; i++){
    pinMode(ledPins[i], OUTPUT);
  }
  Wire.begin();

  lcd.begin(16, 2);
  lcd.clear();

  loadConfigs();
}
```

در تابع loadConfigs مقادیر ذخیره شده در eeprom خوانده شده و در آرایه savedData ذخیره می‌شوند و اگر اولین داده در محدوده مورد نظر برای حالت های ماشین (0 تا 4) بود در currMode ذخیره می‌شود. تابع printLCD برای چاپ حالت صدا زده می‌شود. سپس زمان هایی که برای هر حالت تعیین شده است نیز از 4 داده بعدی خوانده و در آرایه modeTimes ذخیره می‌شوند و با فراخوانی turnOnLED ال‌ای‌دی مورد نظر روشن می‌شود. در نهایت داده ها را در ترمینال چاپ می‌کنیم.

```
void loadConfigs(){
  Serial.println("############ Loading Saved Data ############");
  uint8_t savedData[5];
  eeprom_read(MODE_MEMORY_ADDR, savedData, 5);
  if((char)savedData[0] >= '0' && savedData[0] <= '4')
    currMode = uint8ToInt(savedData[0]);


  printLCD(currMode);

  for(int i=1; i<=4; i++){
      if((char)savedData[i] > '0'){
        modeTimes[i-1] = uint8ToInt(savedData[i]);
      }
  }
  turnOnLED(currMode);

  Serial.println("Current Mode is {" + modeMessages[currMode] + "}");
  Serial.println("Mode Times:");
  for(int i=0; i<FINISH; i++){
      Serial.println(modeMessages[i]+ "{" +  modeTimes[i] + "}" );
  }

}
```

در تابع printLCD ، اگر currMode در محدوده تعریف شده باشد، نام مربوط به آن در LCD چاپ می‌شود. در isKeyNumber چک می‌شود که کاراکتری که در کیپد گرفته شده عدد باشد. تابع turnOnLED عدد حالت را گرفته و بر اساس آن یکی از LED ها که برای آن حالت تعریف شده را روشن و بقیه را خاموش می‌کند. اگر کار ماشین به پایان رسیده باشد و در حالت FINISH باشیم همه روشن می‌شوند. تابع charToInt یک کاراکتر می‌گیرد و با استفاده از کد اسکی به عدد تبدیل می‌کند.

```
void printLCD(int currMode){
  if(currMode > -1){
    lcd.setCursor(0, 0);
    lcd.print("                ");
    lcd.setCursor(0, 0);
    lcd.print(modeMessages[currMode]);
  }
}
bool isKeyNumber(char key){
  int keyAsciiCode = (int)key;
  return keyAsciiCode >= (int)'0' && keyAsciiCode <= (int)'9';
}
void turnOnLED(int currMode){
  if(currMode == FINISH){
    for(int i=PRE_WASH; i<FINISH; i++){
      digitalWrite(ledPins[i], HIGH);
    }
  }else{
    for(int i=PRE_WASH; i<FINISH; i++){
      digitalWrite(ledPins[i], LOW);
    }
    digitalWrite(ledPins[currMode], HIGH);
  }
}
int charToInt(char key){
  return (int)key - 48;
}
```

تابع uint8toInt یک عدد از نوع uint8 گرفته و به int تبدیل کرده و برمی‌گرداند. تابع readKeypad کاراکتری را که وارد شده چک می‌کند و اگر عدد بود در lastKey ریخته و به عنوان کاراکتر مورد نظر نگه می‌دارد. اگر کاراکتر - وارد شود آخرین عددی که وارد شده در EEPROM باید ذخیره شود. ابتدا چک می‌شود که عدد از 4 کمتر باشد. پس از آن کارکاکتر به عدد تبدیل شده و در currMode ذخیره می‌شودو ال‌ای‌دی مورد نظر نیز روشن می‌شود. سپس eeprom_write را فراخوانی می‌کنیم که آرگومان اول آدرس، آرگومان دوم داده و آرگومان سوم size است که اینجا 1 است. حالت ذخیره شده در LCD چاپ می‌شود. اگر کاراکتر C وارد شود یعنی کاربر قبل آن، زمان مورد نظر خود برای آن حالت را وارد کرده است. پس با تبدیل کاراکتر به عدد، آن را در آرایه modeTimes در عنصر مربوطه ذخیره می‌کنیم و در EEPROM ذخیره می‌کنیم. اگر / وارد شد یعنی کاربر می‌خواهد وضعیت فعلی را نگهداری کند و اگر دوباره فشرده شود باید به حالت اولیه برگردد. پس

کافیست زمانی که در آن حالت می‌ماند و isWashing را به درستی تغییر دهیم. در نهایت اگر * فشرده شود باید مقادیر از حافظه خوانده شوند (با فراخوانی loadConfigs) و سپس isWashing به true تغییر مقدار دهد تا مراحل شروع شود.

```
int uint8ToInt(uint8_t in){
    return in - 48;
}
void readKeypad(){
  char key = keypad.getKey();
  static char lastKey;
  if(key){
    Serial.println(key);
    if(isKeyNumber(key)){
        lastKey = key;
    }
    else if(key == '='){//Save in EEPROM
        if(charToInt(lastKey)>4)
          return;
        currMode = charToInt(lastKey);
        turnOnLED(currMode);
        uint8_t tempData[1] = {(uint8_t)lastKey};//ascii code
        eeprom_write(MODE_MEMORY_ADDR, tempData, 1);
        printLCD(currMode);
        lcd.setCursor(0, 1);
        lcd.print("                ");
    }else if(key == 'C'){//Change Time
        modeTimes[currMode] = charToInt(lastKey);
        uint8_t tempData[1] = {(uint8_t)lastKey};
        eeprom_write(MODE_MEMORY_ADDR + 1 + currMode, tempData, 1);
    }else if(key == '/'){
      remainingTime -= millis() - startTime;
      if(!isWashing)
        isWashing = true;
      else
        isWashing = false;
    }else if(key == '*'){
      loadConfigs();
      isWashing = true;
    }
  }
```

تابع eeprom_write، سه آرگومان آدرس داده، داده و اندازه نوشتن را به عنوان ورودی داده می‌گیرد و با فراخوانی Wire.beginTransmission ارتباط با دادن آدرس دیوایس آغاز شده و طبق پروتکل I2C آدرس با فراخوانی write نوشته شده و سپس در یک حلقه دیتا نوشته می‌شود و ارتباط پایان می‌گیرد. تابع eeprom_read به طور مشابه سه ورودی می‌گیرد. پس از نوشتن آدرس، endTransmission فراخوانی شده و پس از آن requestFrom فراخوانی می‌شود که آرگومان اول آن آدرس و دوم تعداد بایت هاست. سپس در یک حلقه با فراخوانی read داده‌ها خوانده می‌شوند.

```
void eeprom_write(uint8_t memory_address, uint8_t* data, int _size){
  Serial.println("--- Writing ---" + String(memory_address));
  Wire.beginTransmission(DEVICE_ADDRESS);
  Wire.write(memory_address);

  for(int i=0; i<_size; i++){
    Wire.write(data[i]);
  }

  Wire.endTransmission();
}
void eeprom_read(uint8_t memory_address, uint8_t *data, uint8_t _size){

  Wire.beginTransmission(DEVICE_ADDRESS);
  Wire.write(memory_address);
  Wire.endTransmission();

  Wire.requestFrom(DEVICE_ADDRESS, _size);
  for(int i=0; i<_size; i++){
    data[i] = Wire.read();
  }

}
```

در تابع checkTime متغیر استاتیک lastMode را برای نگهداری آخرین حالت تعریف می‌کنیم. اگر هنوز فرآیند شروع نشده باشد startTime با millis مقداردهی شده و از تابع خارج می‌شود. اگر lastMode و curMode برابر نباشند مقدار currMode در lastMode ذخیره شده و زمان محاسبه می‌شود. اگر به پایان رسیده بودیم isWashing، false شده و از تابع خارج می‌شود. اگر هیچکدام از این حالات نبود باید زمان محاسبه شود. passedTime زمانیست که از ابتدای حالتی که در آن هستیم گذشته و کافیست آن را از کل زمان یعنی remainingtime کم کنیم. زمان در LCD نیز چاپ می‌شود. اگر زمان کافی سپری شد currMode به حالت بعدی می‌رود و نتایج چاپ و در حافظه ذخیره می‌شوند.

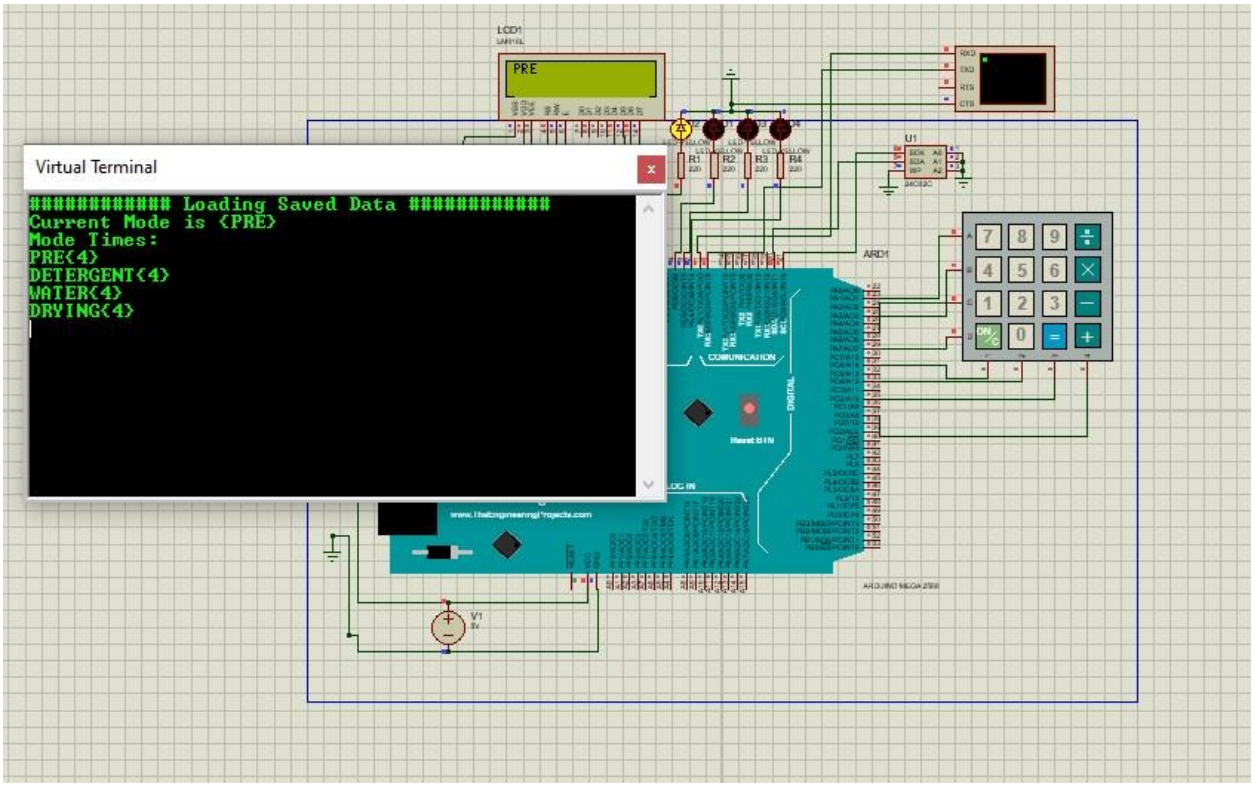
در loop ابتدا readKeypad برلی گرفتن ورودی‌ها و سپس checkTime برای محاسبه حالات و زمان‌های مختلف فراخوانی می‌شود.

```
void checkTime(){
  static int lastMode = -1;
  int passedTime = 0;
  if(!isWashing){
    startTime = millis();
    return;
  }
  if(lastMode != currMode){
    lastMode = currMode;
    startTime = millis();
    remainingTime = modeTimes[currMode] * 1000;
  }else if(currMode == FINISH){
    lastMode = -1;
    isWashing = false;
    loadConfigs();
    return;
  }else{
    int tempTime= (millis() - startTime) % 1000;
    if(tempTime <= 10 || tempTime >= 990){
      passedTime = ((millis() - startTime)/1000);
      //SHOW TIME
      lcd.setCursor(0, 1);
      lcd.print("                ");
      lcd.setCursor(0, 1);
      lcd.print(String(remainingTime/1000  - passedTime)+"s");
    }
    if(remainingTime/1000 -((millis() - startTime)/1000) <= 0){
      currMode = (currMode + 1) % (FINISH+1);
      uint8_t tempData[1] = {currMode + 48};//ascii code
      eeprom_write(MODE_MEMORY_ADDR, tempData, 1);
      printLCD(currMode);
      turnOnLED(currMode);
      lcd.setCursor(0, 1);
      lcd.print("                ");
      if(currMode == 4){
        isWashing = false;
      if(currMode == 4){
        isWashing = false;
      }
    }

  }
}

void loop() {
  readKeypad();
  checkTime();
}
```

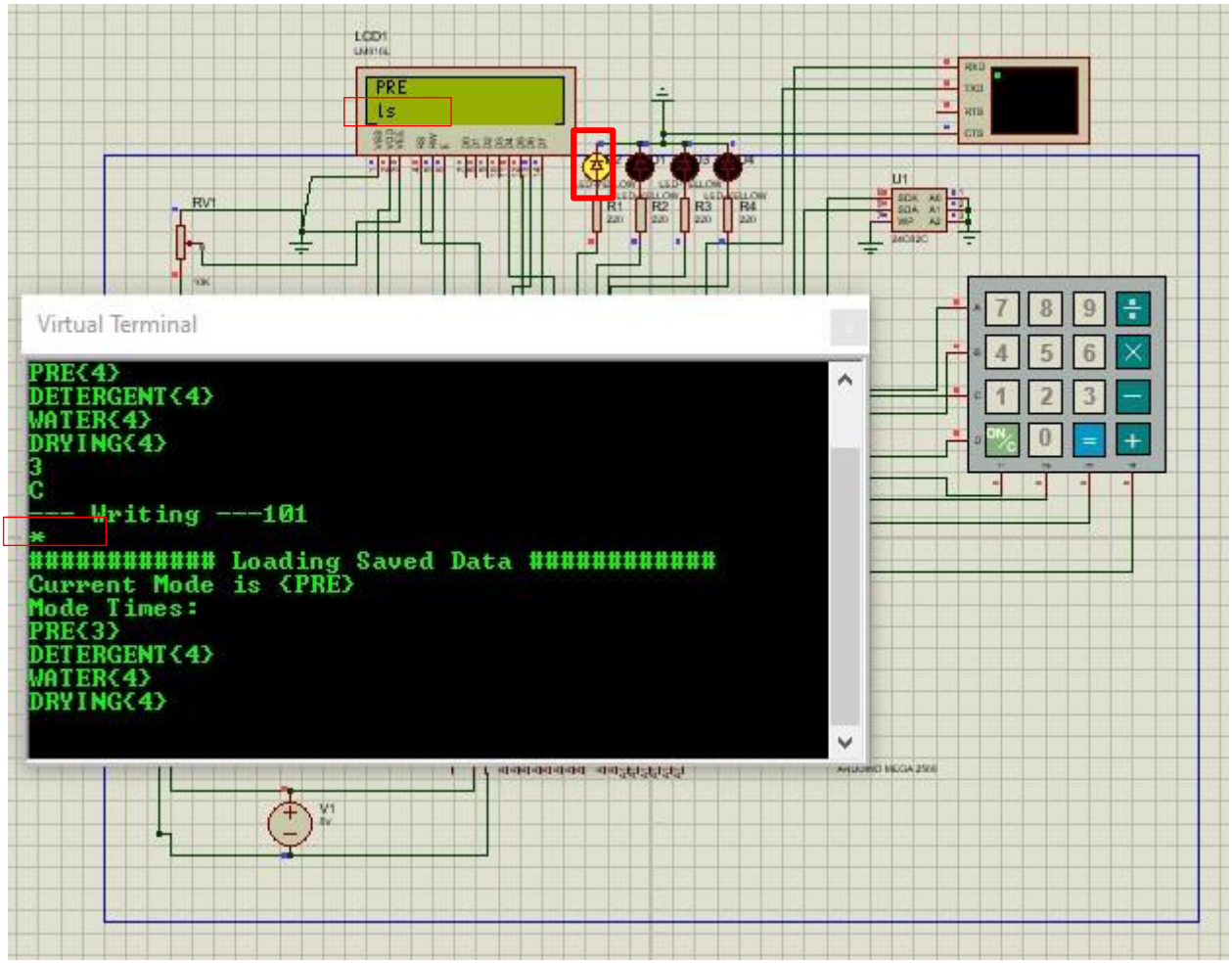
چهار گام کلی

1-پیش شستن 2- شستن با شوینده 3- شستن با آب 4- خشک کردن برای شستن لباس ها انجام می شود.

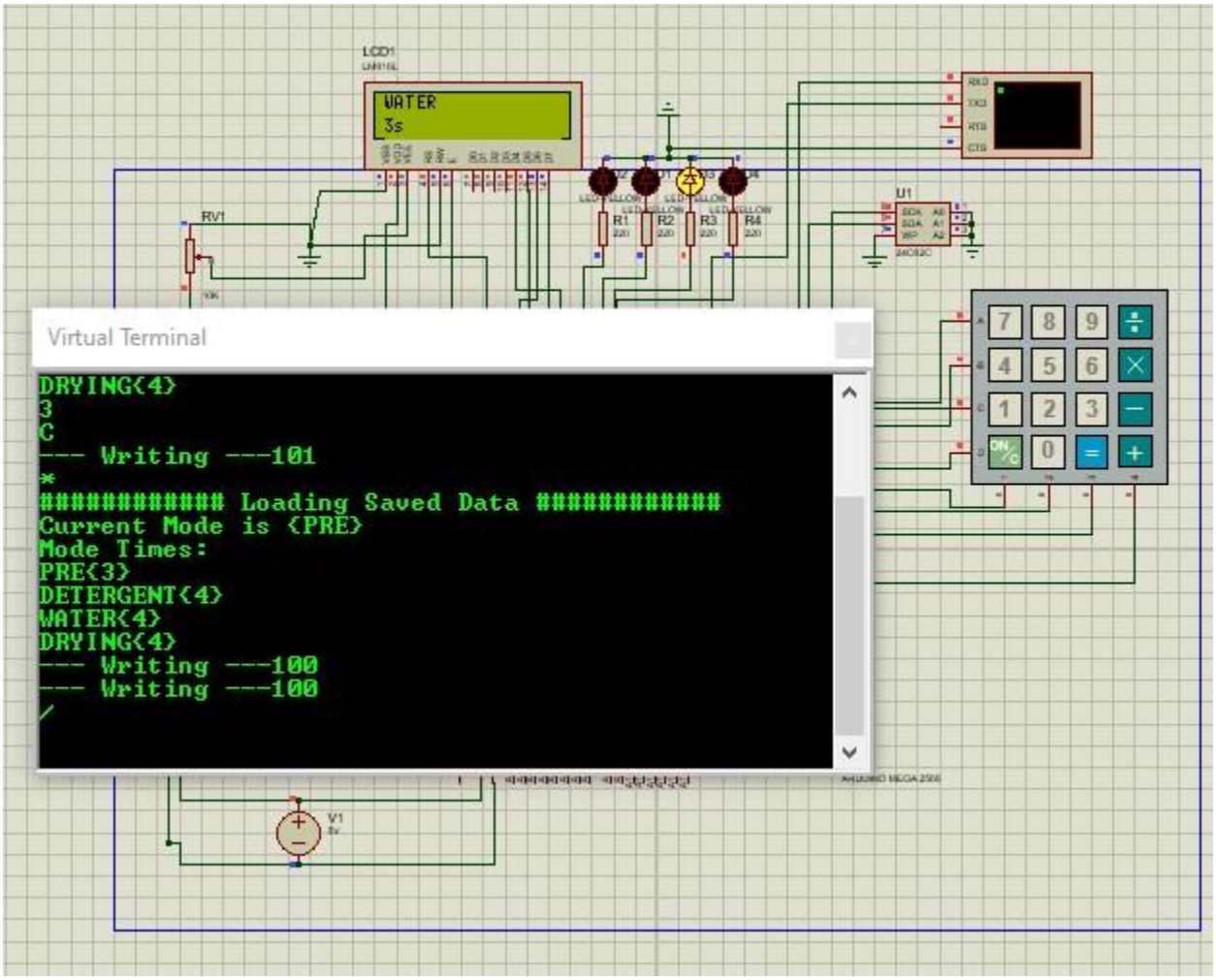
این که دستگاه در کدام هر یک از این گام ها می توانند مدت زمان قابل تنظیم داشته باشند LED .یک از این گام ها است را با روشن بودن یک نشان می دهیم. هر یک از این گام ها می توانند مدت زمان قابل تنظیم داشته باشند.



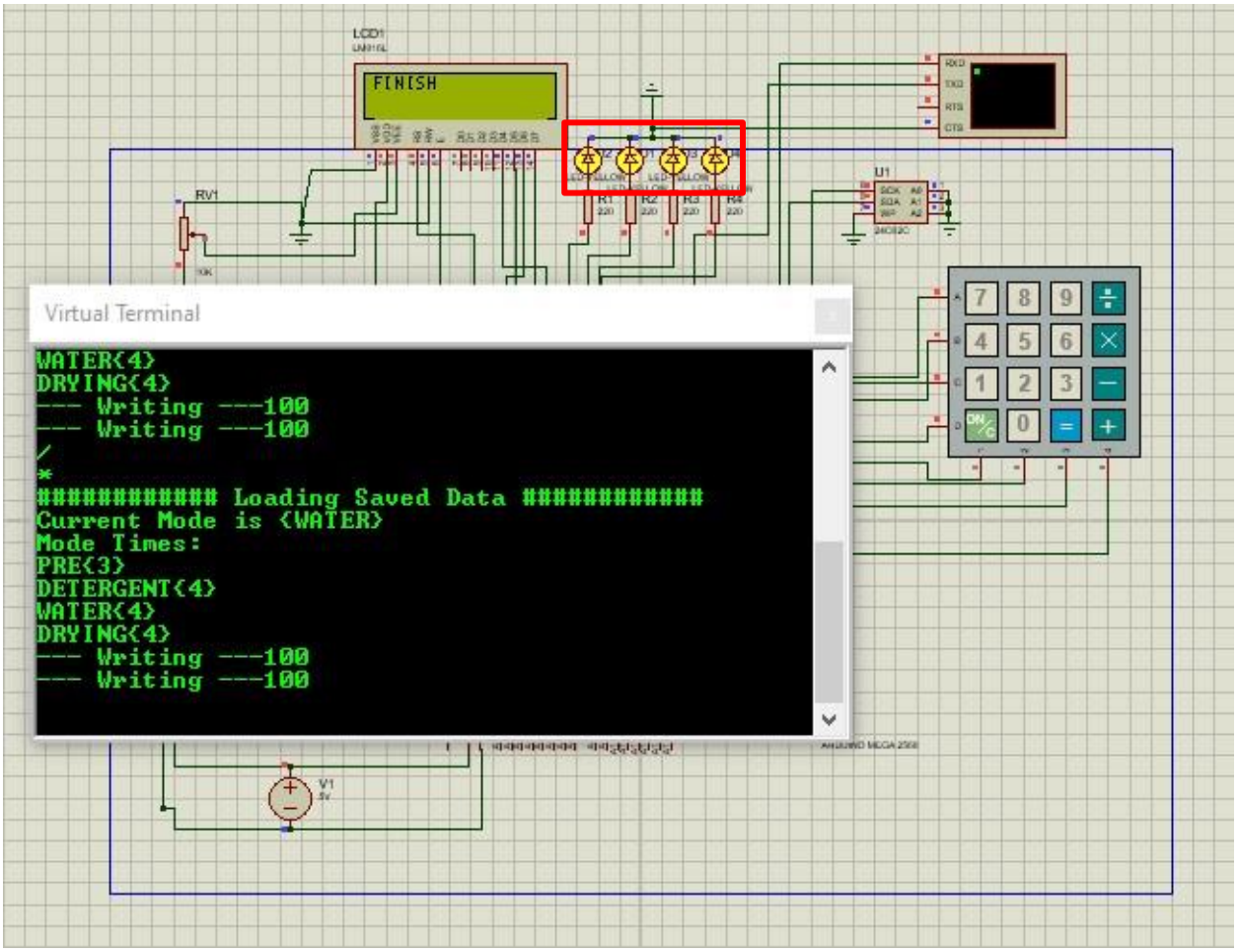
کاربر با زدن c زمان مورد نظر ست شده و با زدن * مقادیر از حافظه خوانده شده و مراحل شروع می شود.



دکمه ی hold دستگاه کاراکتر /است.



در مرحله ی finish تمام ال ای دی ها روشن شده اند.



دستگاه باید مد کاری پیش فرض را بر روی LCD کاراکتری نمایش دهد و هم چنین کاربر باید بتواند مد های دلخواه را با صفحه کاری کلید وارد کند، به گونه ای که با قطع تغذیه نیز در دستگاه ذخیره شده بمانند.