به نام خدا

گزارش آزمایش 2

آزمایشگاه ریزپردازنده و زبان اسمبلی

محمد جواد زندیه، ابوالفضل بکیاسای کیوی

دانشگاه صنعتی امیرکبیر، دانشکده مهندسی کامپیوتر

-انواع keypad ماتریسی و چگونگی کارکرد آنها؟

Keypad های ماتریسی میتوانند 3x4 یا 4x4 (با حتی با تعداد کلید بیشتر) باشند. کلید ها به صورت ماتریسی چیده شده اند به طوری که به ازای هر ردیف یک سیم و به ازای هر ستون نیز یک سیم خواهیم داشت. حال اگر یک دکمه فشرده شود، یکی از سیم های ردیف ها به یکی از سیم های ستون ها متصل شده و بدین صورت میتوان تشخیص داد کدام کلید فشرده شده است (جریان از کدام 2 سیم سطر و ستون میگذرد).

-پدیده ی نوسان (bounce) کلید چیست و چگونه میتوان از بروز اشکالات ناشی از آن جلوگیری کرد؟

پدیده نوسان در سوییچ های مکانیکی اتفاق میافتد و میتواند باعث شود که یک بار فشردن یک دکمه، در سیستم ما بیش از یک بار ثبت شود. این اتفاق به دلیل مکانیکی بودن کلید ها و شامل بودن فنر ها و قطعات فلزی رخ میدهد. برای جلوگیری از این اشکال، یک فاصله زمانی در سیستم نرم افزاری تعیین میشود تا وقتی یک رخداد فشرده شدن کلید تشخصی داده شد، تا گذشت زمان به مقدار آن فاصله مشخص دیگر هیچ رخداد فشردنی ثبت نشود (و اگر تشخیص داده شد نیز در نظر گرفته نشود)

-تعریف مختصر توابع مورد نیاز از کتابخانه Keypad.h:

Keypad(makeKeymap(userKeymap) ،row[] ،col[] ،rows ،cols)

شی مورد نظر برای کنترل Keypad را ساخته و مقادیر اولیه و پین های لازم را در نظر گرفته و مقدار دهی میکند.

char getKey()

اگر کلیدی فشرده شده باشد، کاراکتر معادل با آن را برمیگرداند در غیر این صورت مقدار NO_KEY که همان صفر است را برمیگرداند.

char getKeys()

کلید های فشرده شده به صورت همزمان را برمیگداند

char waitForKey()

پردازش را به طول کامل متوقف میکند تا زمانی که یک کلید فشرده شود. (به جز interrupt routine ها)

KeyState getState()

وضعیت یک کلید را برمیگرداند. یکی از IDLE, PRESSED, RELEASED, HOLD

boolean keyStateChanged()

اگر کلید مورد نظر در آن لحظه تغییر کرده باشد true برمیگرداند و در غیر این صورت false

-مشخصات توابع سریال:

begin()

ارتباط سیگنال را شروع کرده و پین های مربوطه را اشغال میکند. همچنین سرعت انتقال داده را با واحد بیت در ثانیه برای سیگنال مورد نظر مشخص میکند. (برای ترمینال مقدار 9600 استفاده میشود.)

end()

ارتباط سیگنال مورد نظر را تمام کرده و پین ها را آزاد میکند (برای استفاده عمومی به عنوان ورودی و خروجی)

find()

یک رشته به عنوان ورودی میگیرد و تا جایی که به آن رشته نرسیده باشد، از سیگنال داده میخواند. مگر اینکه به محدودیت زمانی خود برسد و false برمیگرداند. اگر به رشته مورد نظر رسید true برمیگرداند.

parseInt()

در سیگنال مربوطه دنبال اولین عدد صحیح قابل خواندن میگردد و به صورت int آنرا برمیگرداند.

read()

یک بایت از داده ی ورودی سیگنال را میخواند و برمیگرداند

readStringUntill()

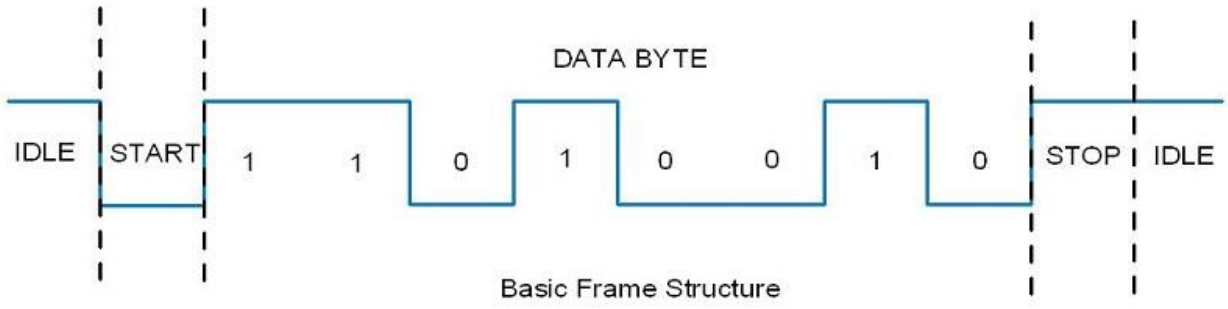
یک کاراکتر دریافت کرده و تا جایی که به آن نرسد، از سیگنال داده را بایت بایت خوانده و در یک string ذخیره کرده و به ما برمیگرداند

write()

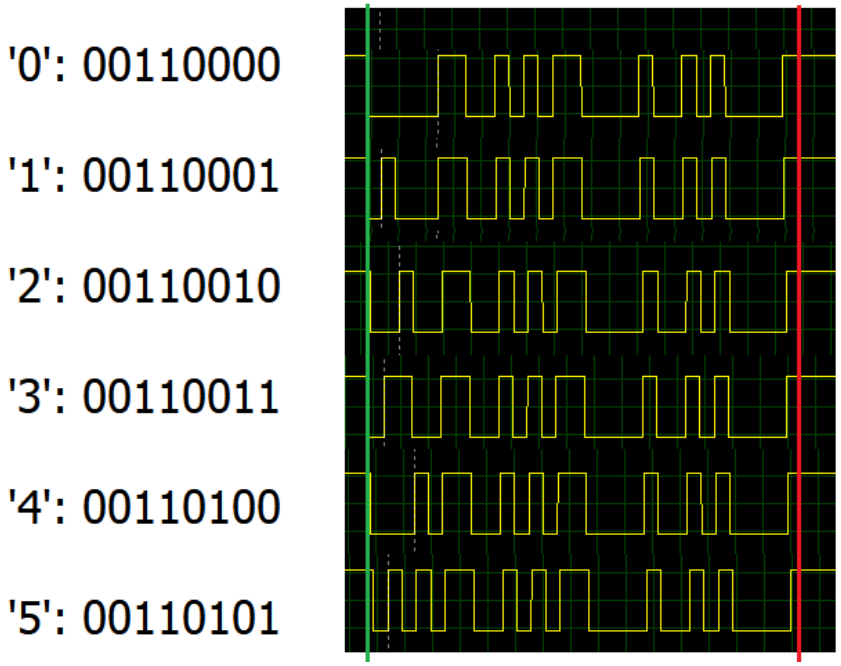
داده ی باینری را به صورت بایت بایت به سیگنال میفرستد.

-پرسش: در رابطه با نحوه ی عملکرد ارتباط سلایر و در اردوینو و همچنین پروتکل های ارتباطی UART و USART توضیح مختصری ارائه دهید.

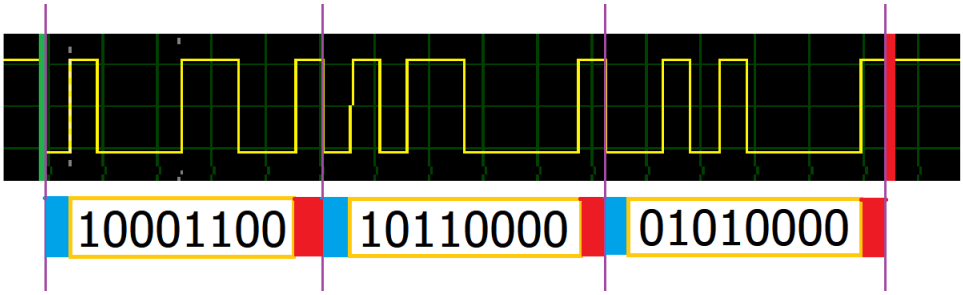
UART مخفف Universal Asynchronous Receiver/Transmitter میباشد (فرستنده/گیرنده ناهمگام جهانی) و USART مخفف (Universal Serial Asynchronous Receiver Transmitter). یک پروتکل جهانی برای ارتباط سریال بین دو دستگاه میباشد. به صورتی که پین Tx هر دستگاه به پین Rx دستگاه دیگر متصل میشود (Tx فرستنده و Rx گیرنده). این مکانیزم با تبدیل داده به یکسری بسته آنها را ارسال میکند و از طرفی دیگر این بسته ها را به داده ها تبدیل میکند. این پروتکل تنظیماتی همچون baud rate, data length, parity bit, number of stop bits, flow control را پشتیبانی میکند. هر بایت در این نوع از انتقال داده، بین 2 بیت شروع و پایان قرار میگیرد. بیت شروع همواره 0 و بیت پایان همواره 1 است.



-سیگنال فرستاده شده به ترمینال روی اسیلوسکوپ را ببینید. آیا میتوانید آن را بررسی کنید؟



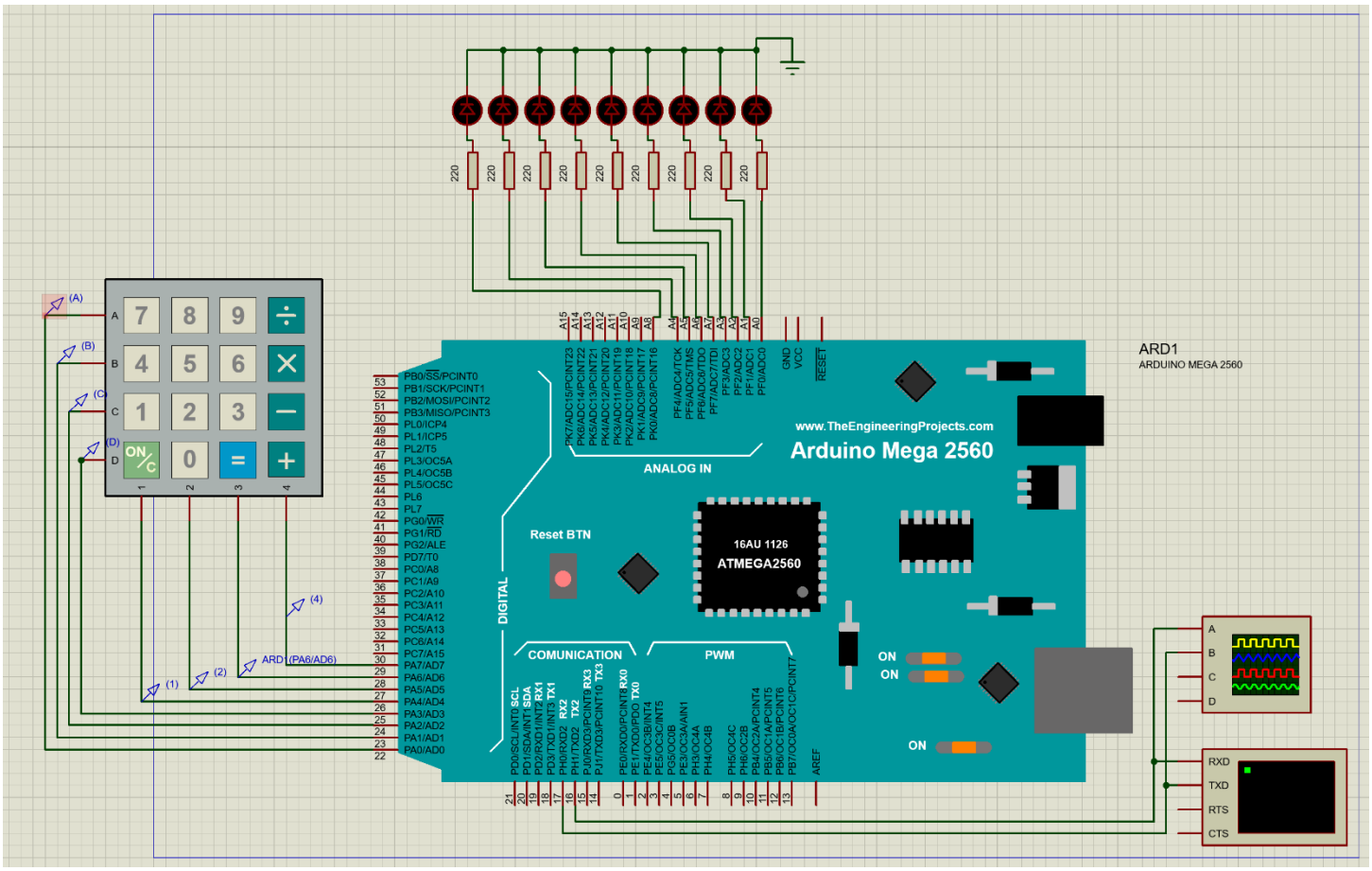
سیگنال های مربوط به کاراکتر های '0' تا '5' را مشاهده میکنیم. در ادامه سیگنال '1' را دقیق تر بررسی می کنیم.



همانطور که در قسمت قبل گفته شد، در UART قبل و بعد از هر بایت از داده یک بیت شروع و یک بیت پایان داریم. اینجا در زیر سیگنال بلاک های هر بایت و بیت های شروع (نوار آبی) و پایان (نوار قرمز) مشخص شده اند. همچنین باید توجه کرد که ما 1 کاراکتر میخواستیم بفرستیم ولی چون از println استفاده شده بود، خود نرم افزار برای ما کاراکتر های \r و \n را نیز بعد از کاراکتر 1 فرستاده است.

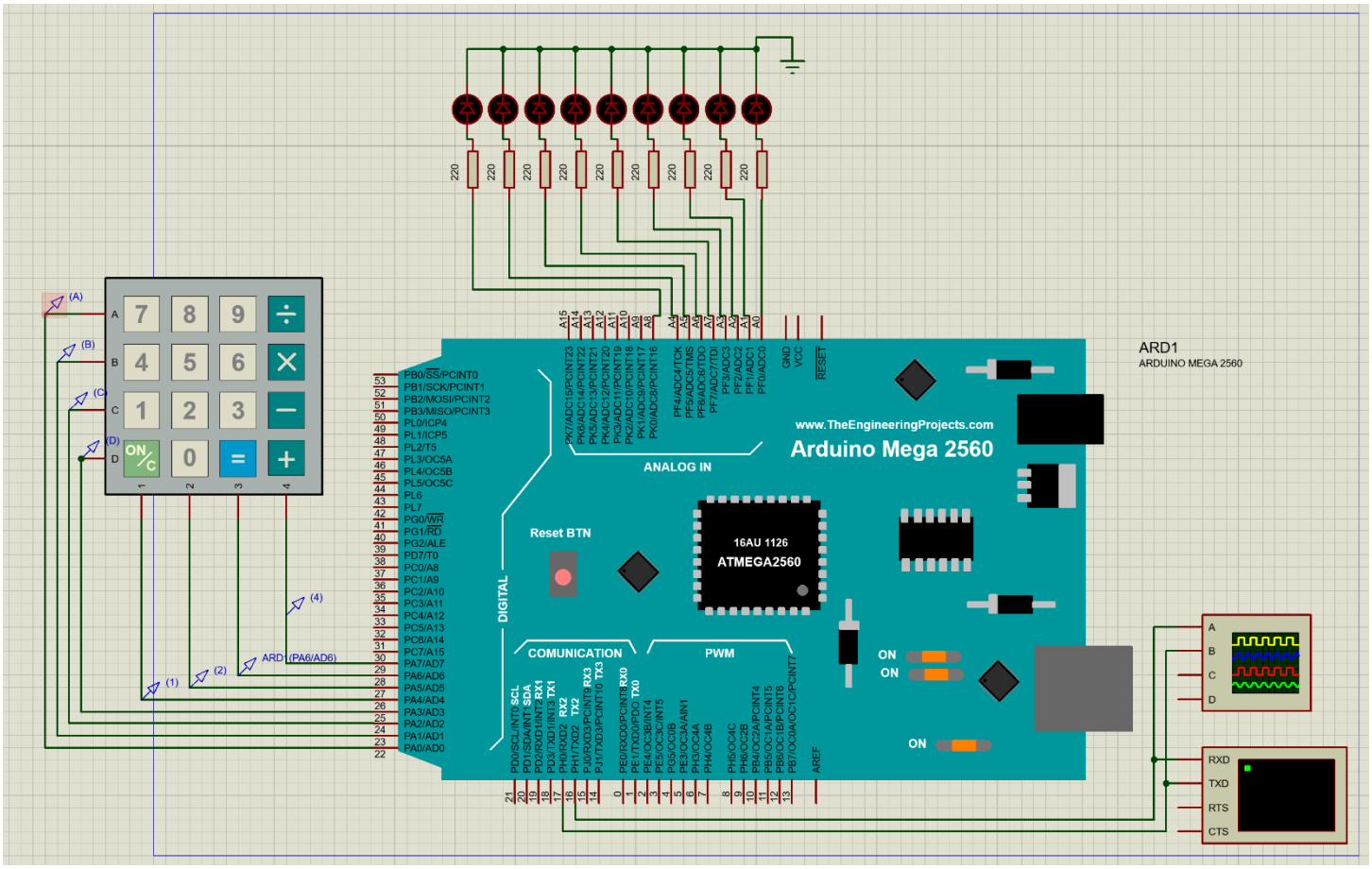
‘1’	Charater 1	49	00110001
‘\r’	Carriage Return	13	00001101
‘\n’	End of Line	10	00001010

همانطور که مشاهده میکنید، بلوک ها از سمت چپ به راست به ترتیب درستی قرار گرفته اند، ولی ترتیب بیت ها داخل هر بایت برعکس شده است (ابتدا بیت های با ارزش کمتر در سیگنال فرستاده شده است)



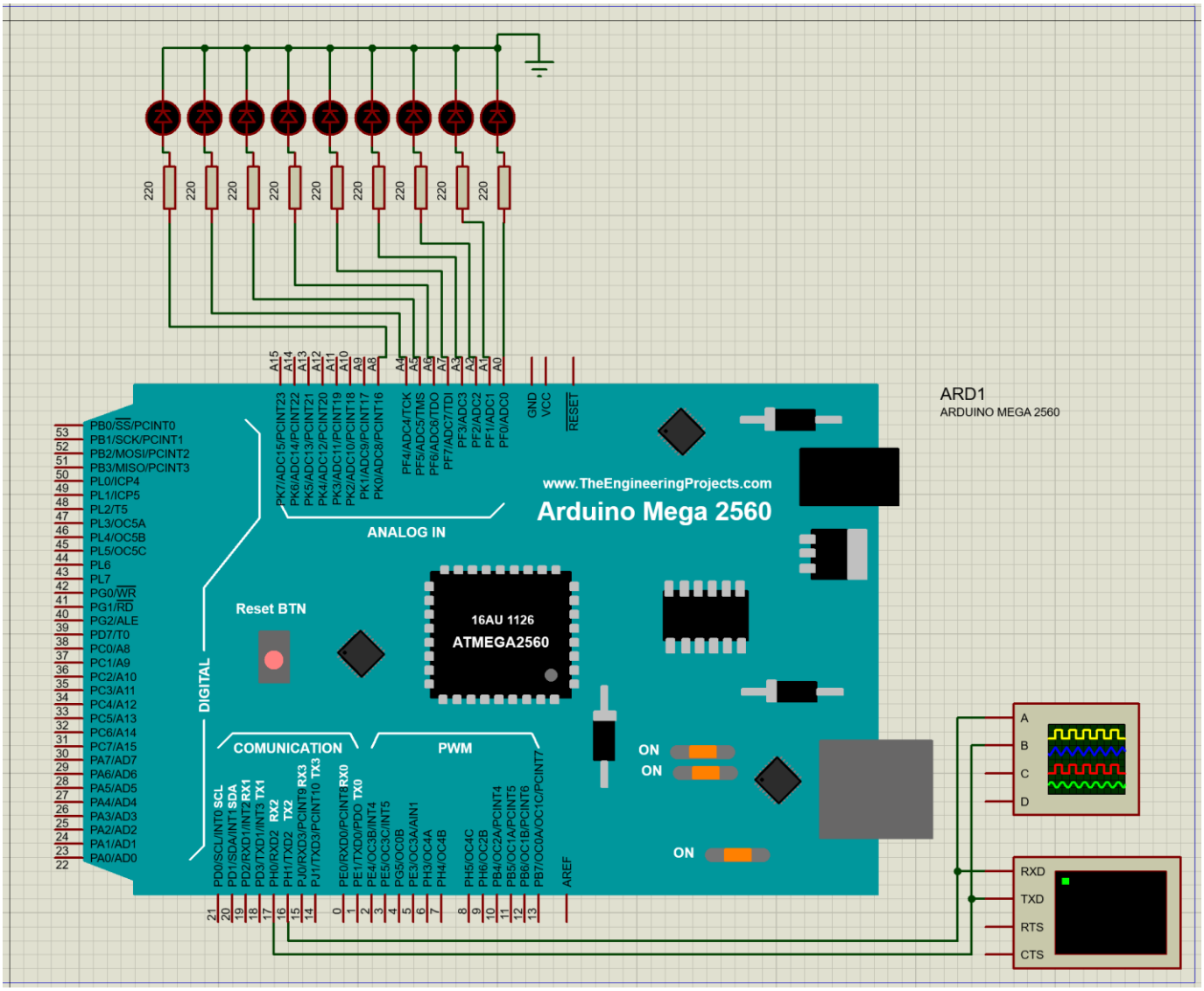

```
// Source code for Problem 1

#include <Keypad.h>

const int LED_NUM = 9;
const int leds[LED_NUM] = {A8, A4, A5, A6, A7, A3, A2, A1, A0};

const byte ROWS = 4;
const byte COLS = 4;
char keys[ROWS][COLS] = {
  {'7','8','9','/'},
  {'4','5','6','*'},
  {'1','2','3','-'},
  {'C','0','=','+'}
};

byte rowPins[ROWS] = {22, 23, 24, 25};
byte colPins[COLS] = {26, 27, 28, 29};
```

```
Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);

int LED_ON_Num = 0;

void setup(){
  for(int i=0;i<LED_NUM;i++)
    pinMode(leds[i], OUTPUT);
  keypad.addEventListener(keypadEvent);
}

void loop(){
  for(int i=0;i<LED_NUM;i++)
    digitalWrite(leds[i], (i < LED_ON_Num ? HIGH : LOW));
  char key = keypad.getKey();
}
void keypadEvent(KeypadEvent key) {
  int n = key - '0';
  if(n < 0 || n > LED_NUM)
    return;
  switch (keypad.getState()) {
    case PRESSED:
      LED_ON_Num = n;
    break;
    case RELEASED:
      //LED_ON_Num = 0;
    break;
  }
}
```




```
// Source code for Problem 2

#include <Keypad.h>

const int LED_NUM = 9;
const int leds[LED_NUM] = {A8, A4, A5, A6, A7, A3, A2, A1, A0};

const byte ROWS = 4;
const byte COLS = 4;
char keys[ROWS][COLS] = {
  {'7','8','9','/'},
  {'4','5','6','*'},
  {'1','2','3','-'},
  {'C','0','=','+'}
};

byte rowPins[ROWS] = {22, 23, 24, 25};
byte colPins[COLS] = {26, 27, 28, 29};
```

```
Keypad keypad = Keypad(makeKeymap(keys), rowPins, colPins, ROWS, COLS);

int LED_ON_Num = 0;

void setup(){
  keypad.addEventListener(keypadEvent);
  Serial2.begin(9600);
}

void loop(){
  char key = keypad.getKey();
}

void keypadEvent(KeypadEvent key) {
  if(keypad.getState() == PRESSED)
    Serial2.println(key);
}
```




```
// Source code for Problem 3

const int LED_NUM = 9;
const int leds[LED_NUM] = {A8, A4, A5, A6, A7, A3, A2, A1, A0};

int LED_ON_Num = 0;

void setup(){
  for(int i=0;i<LED_NUM;i++)
    pinMode(leds[i], OUTPUT);
  Serial2.begin(9600);
}
```

```
void loop(){
  for(int i=0;i<LED_NUM;i++)
    digitalWrite(leds[i], (i < LED_ON_Num ? HIGH : LOW));
  if(Serial2.available() > 0) {
    int num = Serial2.parseInt();
    Serial2.println(num);
    if(num < 0 || num > 9)
      Serial2.println("Invalid number");
    else
      LED_ON_Num = num;
  }
}
```