به نام خدا

پیش گزارش آزمایش دوم آزمایشگاه ریز پردازنده

امیرپارسا سلمان خواه

۹۸۳۱۰۳۴

پرسش: در رابطه با نحوه ی عملکرد ارتباط سریال در آردوییِنو و همچنین پروتکل های ارتباطی UART و USART توضیح مختصری ارایه دهید.

ارتباط های Serial برای انتقال داده ها به شکل سری استفاده می شوند. این روش در مقابل با روش Parallel قرار دارد که با استفاده از چندین سیم و به طور همزمان بیت ها را از یک دستگاه به دستگاه دیگر انتقال می دهد.

ارتباط های Serial انواع مختلفی دارند که از مهم ترین آن ها می توان به UART و I2C و SPI اشاره کرد.

در ارتباط UART کلاک وجود ندارد و برقراری ارتباط به شکل asynchronous است. برای برقراری چنین ارتباطی لازم است تا گیرنده و فرستنده بر سر ۳ چیز توافق داشته باشند: سرعت ارسال اطلاعات، بیت های آغاز و پایان و طول داده. به این شکل گیرنده با توجه به این اطلاعات می تواند متوجه شود که فرستنده در چه زمانی شروع به ارسال اطلاعات کرده و چه اطلاعاتی را ارسال کرده است.

در ارتباط USART کلاک وجود دارد و برقراری ارتباط به شکل synchronous صورت می گیرد. به این شکل که در لبه ی کلاک فرستنده داده ها را می فرستد و گیرنده آن را دریافت می کند. این روش نسبت به روش UART کمتر استفاده می شود.

در آردوییِنو به کمک پایه های Tx و Rx می توان از UART و USART استفاده کرد. از پایه ی Tx برای ارسال اطلاعات و از پایه ی Rx برای دریافت اطلاعات استفاده می شود.

انواع کیبورد های ماتریسی و چگونگی کارکرد آن ها

به طور کلی دو نوع کیپد ماتریسی داریم. کیپد معمولی و خازنی.

کیپد معمولی به این شکل است که که با فشردن یک کلید، اتصال بین سیم های مربوط به ردیف و ستون آن کلید برقرار می شود و با توجه به آن متوجه می شویم که کدام دکمه فشرده شده است.

کیپد خازنی به این شکل است که با فشرده شدن کلید، یک خازن مربوط به آن کلید شارژ می شود و با توجه به آن خازن فشرده شدن کلید مشخص می شود.

پدیده ی نوسان کلید چیست و چگونه می توان از بروز آن جلوگیری کرد؟

هنگامی که کلیدی را فشار می دهیم، تا زمانی که دو سمت کلید کاملا به هم متصل نشده اند، بخش های مختلف فلز استفاده شده در ساخت کلید بار ها به یکدیگر وصل و

از یکدیگر قطع می شوند. به این پدیده نوسان کلید می گویند. این پدیده موجب می شود که ورودی ما چندین بار به اشتباه مقدار ۱ منطقی را دریافت کند. البته این پدیده هنگام قطع کردن کلید نیز اتفاق می افتد.
یک راه برای جلوگیری از این اتفاق، **اضافه کردن یک خازن به مدار** است به این صورت که سر منفی آن به زمین و سر دیگر آن به محل پین ورودی میکروکنترلر وصل می شود. در این صورت نوسانات ابتدایی ناشی از وصل شدن کلید موجب شارژ شدن خازن می شود و وارد میکروکنترلر نمی شود تا زمانی که خازن پر شود و در این صورت خازن در حالت مدار باز قرار میگیرد و ورودی وارد میکروکنترلر می شود.
راه دیگر **استفاده از SR Latch** است. در این روش هنگامی که کلید باز است، پایه ی R لچ مقدار یک را دارد و خروجی مقدار صفر را خواهد داشت. هنگامی که کلید بسته می شود، ابتدا مقدار پایه ی R صفر می شود و لچ در حالت hold قرار می گیرد. پس فعلا همان مقدار صفر را در خروجی خواهیم داشت. سپس کلید باعث می شود تا پایه ی S برابر با یک شود. در این صورت مقدار خروجی برابر با یک می شود. حال اگر bouncing اتفاق بیفتد مقدار پایه ی S بین صفر و یک نوسان می کند و این باعث می شود که لچ از حالت Set به Hold و برعکس منتقل شود که در این صورت مقدار خروجی همواره برابر با یک خواهد بود و دیگر در خروجی bouncing را شاهد نخواهیم بود.
راه سوم **استفاده از روش های نرم افزاری** برای جلوگیری از این اتفاق است. به عنوان مثال کتابخانه ی keypad.h که از آن برای برقراری ارتباط با کیپد استفاده می کنیم، از delay برای debouncing استفاده می کند.

تعریف مختصر توابع مورد نیاز از کتابخانه ی keypad.h

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

این تابع یک شی از نوع Keypad می سازد.
در آرگومان اول باید با کمک تابع makekeymap دکمه های ورودی را برای آن تعریف کنیم. این تابع یک آرایه ی دو بعدی از کاراکترها را به عنوان ورودی دریافت می کند که در واقع دکمه های ما هستند.
در آرگومان دوم و سوم، یک لیست از پین های مربوط به سطر ها و ستون ها داده می شود.
در آرگومان چهارم و پنجم نیز تعداد سطر ها و ستون های کیپد داده می شود.

```
2)Char getKey()
```

این تابع دکمه ای که فشرده شده است را بر میگرداند.

```
3)boolean getKeys()
```

این تابع اگر دکمه ای فعال شد، مقدار true و در غیر این صورت مقدار false را بر می گرداند.

```
4)Char waitForKey()
```

این تابع تا بی نهایت صبر می کند تا یک دکمه فشرده شود و آن را بر می گرداند.

```
5)KeyState getState()
```

این تابع وضعیت همه ی کلید های کیپد را بر می گرداند. این وضعیت ها شامل مقادیر زیر هستند:

```
IDLE, PRESSED, HOLD, RELEASED
```

```
6)boolean keyStateChanged()
```

این تابع مشخص می کند آیا وضعیت کی پد تغییر کرده است یا خیر. به بیان دیگر هرگاه KeyState تغییر کند این تابع مقدار true را بر می گرداند.

نحوه و کاربرد های ارتباط سریال در آردوییِنو

در مورد نحوه ی ارتباطات سریال در پرسش اول صحبت شد.
از ارتباطات سریال برای تعامل با سایر کامپیوتر ها یا میکروکنترلر ها و یا ماژول های آن ها استفاده می شود. حال نحوه ی برقراری این ارتباط توسط آن ماژول یا میکروکنترلر تعیین می شود. مثلا ممکن است یک ماژول از روش I۲C و ماژولی دیگر از روش SPI برای برقراری ارتباط استفاده کند.
این روش ها نسبت به یکدیگر مزایا و معایبی دارند. مثلا در روش UART می توانیم فقط با یک دستگاه دیگر ارتباط برقرار کنیم. در حالی در دو روش دیگر می توانیم با چند دستگاه ارتباط برقرار کنیم.

تعریف مختصر و نحوه ی کار با توابع ارتباط سریال

`begin()`

این تابع baud rate یا سرعت انتقال داده را مشخص می کند.

`end()`

ارتباط سریال را قطع می کند.

`find()`

تا زمانی که دستگاه دوم پیدا شود، از پورت سریال داده می خواند و در صورت پیدا شدن مقدار true را بر می گرداند.

`parseInt()`

اولین عدد صحیح معتبر را در سریال ورودی پیدا می کند. البته این کار را تا زمان timeout شدن انجام می دهد.

`println()`

دیتا را به شکل ASCII روی سریال خروجی می ریزد.

`read()`

دیتای ورودی را می خواند.

`readStringUntil()`

کاراکتر های سریال ورودی را به شکل استرینگ می خواند.

`write()`

روی سریال خروجی به شکل بایت دیتا می ریزد.

کد بخش اول:

```
#include <Keypad.h>

const byte ROWS = 4;
const byte COLS = 4;

char keys[ROWS][COLS] = {
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};

#define LED1 5
#define LED2 6
#define LED3 7
#define LED4 8
#define LED5 9
#define LED6 10
#define LED7 11
#define LED8 12
#define LED9 13

#define KEYPAD1 A5
#define KEYPAD2 A4
```

```
#define KEYPAD3 A3

#define KEYPAD4 A2

#define KEYPAD5 A1

#define KEYPAD6 A0

#define KEYPAD7 2

#define KEYPAD8 3



byte rowPins[ROWS] = {KEYPAD1, KEYPAD2, KEYPAD3, KEYPAD4}; //connect to the
row pinouts of the keypad

byte colPins[COLS] = {KEYPAD5, KEYPAD6, KEYPAD7, KEYPAD8}; //connect to the
column pinouts of the keypad



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



void setup() {

  // put your setup code here, to run once:

  pinMode(LED1, OUTPUT);

  pinMode(LED2, OUTPUT);

  pinMode(LED3, OUTPUT);

  pinMode(LED4, OUTPUT);

  pinMode(LED5, OUTPUT);

  pinMode(LED6, OUTPUT);

  pinMode(LED7, OUTPUT);

  pinMode(LED8, OUTPUT);

  pinMode(LED9, OUTPUT);
```

```
  Serial.begin(9600);
}

void loop() {
  // Getting the selected key
  char key = keypad.getKey();

  // Check if the key is a number
  if (key != NO_KEY && key >= '1' && key <= '9'){
    int number = (int) (key - '0');
    // Turn on LEDs
    for(int i=LED1; i < LED1 + number; i++){
      digitalWrite(i, HIGH);
    }
    delay(2000);
    // Turn off LEDs
    for(int i=LED1; i < LED1 + number; i++){
      digitalWrite(i, LOW);
    }
  }
}
```

کد بخش دوم:

```
#include <Keypad.h>

const byte ROWS = 4;
const byte COLS = 4;

char keys[ROWS][COLS] = {
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};

#define LED1 5
#define LED2 6
#define LED3 7
#define LED4 8
#define LED5 9
#define LED6 10
#define LED7 11
#define LED8 12
#define LED9 13

#define KEYPAD1 A5
#define KEYPAD2 A4
```

```
#define KEYPAD3 A3

#define KEYPAD4 A2

#define KEYPAD5 A1

#define KEYPAD6 A0

#define KEYPAD7 2

#define KEYPAD8 3



byte rowPins[ROWS] = {KEYPAD1, KEYPAD2, KEYPAD3, KEYPAD4}; //connect to the
row pinouts of the keypad

byte colPins[COLS] = {KEYPAD5, KEYPAD6, KEYPAD7, KEYPAD8}; //connect to the
column pinouts of the keypad



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



void setup() {

  // put your setup code here, to run once:

  pinMode(LED1, OUTPUT);

  pinMode(LED2, OUTPUT);

  pinMode(LED3, OUTPUT);

  pinMode(LED4, OUTPUT);

  pinMode(LED5, OUTPUT);

  pinMode(LED6, OUTPUT);

  pinMode(LED7, OUTPUT);

  pinMode(LED8, OUTPUT);

  pinMode(LED9, OUTPUT);
```

```
  Serial.begin(9600);
}

void loop() {
  // Getting the selected key
  char key = keypad.getKey();

  // Print the key
  if (key != NO_KEY){
    Serial.println(key);
  }
}
```

کد بخش سوم:

```
#include <Keypad.h>

const byte ROWS = 4;
const byte COLS = 4;

char keys[ROWS][COLS] = {
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};

#define LED1 5
#define LED2 6
#define LED3 7
#define LED4 8
#define LED5 9
#define LED6 10
#define LED7 11
#define LED8 12
#define LED9 13

#define KEYPAD1 A5
#define KEYPAD2 A4
```

```
#define KEYPAD3 A3

#define KEYPAD4 A2

#define KEYPAD5 A1

#define KEYPAD6 A0

#define KEYPAD7 2

#define KEYPAD8 3



byte rowPins[ROWS] = {KEYPAD1, KEYPAD2, KEYPAD3, KEYPAD4}; //connect to the
row pinouts of the keypad

byte colPins[COLS] = {KEYPAD5, KEYPAD6, KEYPAD7, KEYPAD8}; //connect to the
column pinouts of the keypad



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



int number = 0;



void setup() {

  // put your setup code here, to run once:

  pinMode(LED1, OUTPUT);

  pinMode(LED2, OUTPUT);

  pinMode(LED3, OUTPUT);

  pinMode(LED4, OUTPUT);

  pinMode(LED5, OUTPUT);

  pinMode(LED6, OUTPUT);

  pinMode(LED7, OUTPUT);

  pinMode(LED8, OUTPUT);
```

```
  pinMode(LED9, OUTPUT);

  Serial.begin(9600);

  Serial.print("Please enter your number: ");
}

void loop() {
  // Getting the selected key
  char key = keypad.waitForKey();
  // Print the key
  if (key >= '0' && key <= '9'){
    if (number != 0 || key != '0'){
      Serial.print(key);
    }
    int input = (int) (key - '0');
    number = number * 10 + input;
  }
  else if (key == 'D'){
    if (number <= 9){
      // Turn on LEDs
      for(int i=LED9; i > LED9 - number; i--){
        digitalWrite(i, HIGH);
      }
      delay(2000);
```

```
      // Turn off LEDs
      for(int i=LED9; i > LED9 - number; i--){
        digitalWrite(i, LOW);
      }

      Serial.println("\nDone!");
    }
    else{
      Serial.println("\nInvalid input!");
    }
    Serial.print("Please enter your number: ");
    number = 0;
  }
}
```