شیلد نرم افزاری Timer

تابع delay در آردینو جهت ایجاد تاخیر زمانی استفاده می شود ولی وقتی این تابع در اجرای برنامه تاخیر ایجاد می کند هیچ عمل دیگری مانند به روز رسانی نمایشگر یا تشخیص کلید های فشرده شده نمی تواند صورت پذیرد. نمونه کد زیر جهت روشن و خاموش کردن یک led با استفاده از تابع delay می باشد

```
int pin = 13;
 void setup()
{
 pinMode(13, OUTPUT);
 digitalWrite(pin, HIGH);
 delay(10 * 60 * 1000);
 digitalWrite(pin, LOW);
}
 void loop()
{
}
```

با استفاده از شیلد نرم افزاری Timer شما میتوانید اعمالی را که نیاز به تاخیر زمانی دارند را بدون این که در اجرای فرامین سایر قسمت ها اخلالی ایجاد کند را انجام دهید این شیلد توسط "Simon Monk" نوشته شده که من تغییراتی در کد آن ایجاد کرده ام تا این شیلد از softShield پیروی کند.

نمونه کدی که در بالا به آن اشاره شد با استفاده از شیلد Timer به این صورت نوشته می شود:

```
#include <softShield.h>
#include <Event.h>
#include <Timer.h>

Timer t;

int pin = 13;

void setup(){
 pinMode(pin, OUTPUT);
 t.pulse(pin, 10 * 60 * 1000, HIGH); // 10 minutes
}
```

```
void loop(){
 softModule::update();
}
```

ورودی های تابع pulse به ترتیب ۱. شماره پایه آردینو جهت تغییر ۲. زمان تغییر ۳. وضعیت اولیه پایه می باشد.

همانطور که قبلا اشاره شد softShield توابع مربوط به update و setup شیلد های نرم افزاری را به صورت اتوماتیک اجرا میکند و همین عامل باعث تشخیص زمان توسط Timer می شود.

در مثال زیر استفاده از Timer برای دو عمل روشن و خاموش کردن led و خواندن پایه A0 آردینو و نمایش آن در Serial Monitor را نمایش داده شده است:

```
#include <softShield.h>
#include <Event.h>
#include <Timer.h>

Timer t;
int pin = 13;

 void setup()
{
 Serial.begin(9600);
 pinMode(pin, OUTPUT);
 t.oscillate(pin, 100, LOW);
 t.every(1000, takeReading);
}


void loop()
{
 softModule::update();
}


void takeReading()
{
 Serial.println(analogRead(0));
}
```

اولین نکته ای که در اینجا باید به آن اشاره کرد این است که ما در این مثال از یک تابع به نام "takeReading" استفاده کرده ایم که این تابع به عنوان ورودی تابع every از شیلد Timer استفاده شده است. این عمل سبب اجرای تابع "takeReading" در هر یک ثانیه می شود.

همچنین با استفاده از تابع "oscillate" یک رویداد دیگر نیز به Timer اضافه کرده ایم که هر ۱۰۰ میلی ثانیه حالت led را عوض می کند.

هرکدام از این رویداد ها دارای یک شناسه (ID) هستد که توسط این شناسه میتوانیم آن رویداد را متوقف کنیم. همانطور که در مثال زیر نشان داده شده است هر ۲ ثانیه عبارتی به پورت سریال فرستاده می شود (توسط تابع doSomething) ، led روشن و خاموش شده و بعد از ۵ ثانیه سرعت آن ۵ برابر کمتر می شود.

```
#include <softShield.h>
#include <Event.h>
#include <Timer.h>

Timer t;

int ledEvent;

void setup()
{
  Serial.begin(9600);
  int tickEvent = t.every(2000, doSomething);
  Serial.print("2 second tick started id=");
  Serial.println(tickEvent);

  pinMode(13, OUTPUT);
  ledEvent = t.oscillate(13, 50, HIGH);
  Serial.print("LED event started id=");
  Serial.println(ledEvent);

  int afterEvent = t.after(10000, doAfter);
  Serial.print("After event started id=");
  Serial.println(afterEvent);
}

void loop()
{
  t.update();
}


void doSomething()
{
  Serial.print("2 second tick: millis()=");
```

```
 Serial.println(millis());
}

 void doAfter()
{
 Serial.println("stop the led event");
 t.stop(ledEvent);
 t.oscillate(13, 500, HIGH, 5);
}
```

هر Timer میتواند شامل حداکثر ۱۰ رویداد مختلف باشد.

توابع شیلد Timer

int every(long period, callback)

تابع callback را در هر period میلی ثانیه اجرا می کند و شناسه مربوط به رویداد را به عنوان نتیجه تابع بر می گرداند.

int every(long period, callback, int repeatCount)

تابع callback را در هر period میلی ثانیه ، repeatCount بار اجرا می کند و شناسه مربوط به رویداد را به عنوان نتیجه تابع بر می گرداند.

int after(long duration, callback)

تابع callback را فقط و فقط یک بار پس از duration میلی ثانیه اجرا می کند و شناسه مربوط به رویداد را به عنوان نتیجه تابع بر می گرداند.

int oscillate(int pin, long period, int startingValue)

حالت پایه pin را در هر period میلی ثانیه از HIGH به LOW و با بلعکس تغییر می دهد و مقدار اولیه پایه توسط startingValue مشخص می شود که باید HIGH و یا LOW باشد و شناسه مربوط به رویداد را به عنوان نتیجه تابع بر می گرداند.

int oscillate(int pin, long period, int startingValue, int repeatCount)

حالت پایه pin را در هر period میلی ثانیه از HIGH به LOW و با بلعکس تغییر می دهد و مقدار اولیه پایه توسط startingValue مشخص می شود که باید HIGH و یا LOW باشد این عمل repeatCount بار تکرار می شود. شناسه مربوط به رویداد را به عنوان نتیجه تابع بر می گرداند.

int pulse(int pin, long period, int startingValue)

حالت پایه pin را در هر period میلی ثانیه از HIGH به LOW و با بلعکس تغییر می دهد و مقدار اولیه پایه توسط startingValue مشخص می شود که باید HIGH و یا LOW باشد این عمل فقط و فقط یک بار تکرار می شود. شناسه مربوط به رویداد را به عنوان نتیجه تابع بر می گرداند.

int stop(int id)

رویداد Timer با شناسه id را متوقف می کند.