ArnoTek Arduino Utilities – HeartBeat

Arduinos have an onboard LED that is connected to pin 13 and this makes a nice heartbeat indicator that the Arduino is running as it should and that it is not stuck in an infinite loop, excessive delay, or is in sleep mode. Also, depending on your application, sometimes it is necessary to invoke code at particular time intervals such as polling. These capabilities are what this heartbeat utility addresses.

To use this utility, install the **ArnoTek_HeartBeat.h** file in your Arduino libraries directory. This will make the utility accessible to all of your projects. ArnoTek utilities and global variables are prefixed with "ArnoTek" in an effort to prevent conflicts with include files from other developers.

The ArnoTek_HeartBeat utility can be installed using the Arduino Library Manager by searching for "ArnoTek" and clicking on its corresponding INSTALL button. You must include the library at the top of your code with the compiler directive to use the utility:

#include <ArnoTek_HeartBeat.h>

Probably the most utilized feature of the ArnoTek_HeartBeat utility is proving a simple heartbeat indicator with minimal user code. However, the HeartBeat utility also has the ability to call a user function on a periodic time interval such invoking a polling routine. The following examples demonstrate some of the capabilities of the HeartBeat utility.

Copyright 2026 Anthony J. Arnovitz All Rights Reserved

ArnoTek Arduino Utilities – HeartBeat

Example 0

This example defines your object and uses the default timeout value of 500 milliseconds and toggles the LED connected to the default pin 13. You need to define an object of the ArnoTek_HeartBeat class, typically before your ***setup*** function.

ArnoTek_HeartBeat HeartBeat;

- "ArnoTek_HeartBeat" is the name of the class.
- "HeartBeat" is the name of the object for your code. You may name this anything that you wish.

Now that your object has been defined, you only need to place one line of code in your ***loop*** function to call the ***Toggle*** method in your class. This is typically placed at the top of your ***loop*** function.

HeartBeat.Toggle();

Every time through the ***loop*** function, the ***Toggle*** function determines if the default number of milliseconds (500) has elapsed and if it has, the LED connected to the default GPIO pin (13) is turned on. After the specified number of milliseconds has passed a second time, the LED is turned off and so on. Thus, the LED toggles on and off at half second intervals.

Copyright 2026 Anthony J. Arnovitz All Rights Reserved

ArnoTek Arduino Utilities – HeartBeat

Example 1

This example defines your object with a timeout value 250 milliseconds and toggles the LED connected to default pin 13. This constructor supports one parameter that is the interval in milliseconds that you wish to toggle the LED. This parameter is a long unsigned variable and it may be a value from 1 millisecond to 4,294,967,295 milliseconds which is 49 days, 17 hours, 2 minutes, 47 seconds, and 296 milliseconds.

ArnoTek_HeartBeat HeartBeat((long unsigned)250);

- "ArnoTek_HeartBeat" is the name of the class.
- "HeartBeat" is the name of the object for your code. You may name this anything that you wish.
- The value of 250 indicates that the LED is to toggle on and off every 250 milliseconds or quarter of a second and is passed as "**long unsigned**" value.

Now that your object has been defined, you only need to place one line of code in your loop function.

HeartBeat.Toggle();

Every time you process the ***loop*** function, the ***Toggle*** function determines if the specified number of milliseconds (250) has elapsed and if it has, the LED connected to the default GPIO pin (13) is turned on. After the specified number of milliseconds has passed a second time, the LED is turned off and so on. Thus, the LED toggles on and off at quarter second intervals.

Copyright 2026 Anthony J. Arnovitz All Rights Reserved

Example 2

But wait, what if you want an additional heartbeat that toggles a different pin for some external component?

You may define as many heartbeat objects as you wish and in this case we will define a second heartbeat object. Say we wish to toggle pin 12 every minute to provide a signal to an external component. The external component would be connected to pin 12 of the Arduino and the object would be defined as

ArnoTek_HeartBeat Pin12_signal((long unsigned)60000, 12);

This constructor supports two parameters, one for the millisecond interval and the second defines the pin to toggle.

Pin12_signal is the name of the object and you may name this anything that you wish. 60000 is the number of milliseconds in a minute and as before, it must be cast as a ***long unsigned*** value. The second parameter indicates the pin to toggle and in this case it is pin 12 but you may use any pin that you wish. Note that there is no verification that the specified pin number is valid. The code in the loop function is identical and in this case is:

HeartBeat.Toggle();

Pin12_signal.Toggle();

The LED at pin 13 toggles every half a second, and pin 12 is toggled every minute. Since the HeartBeat object and the Pin12_signal object are different objects, they are completely independent of each other.

Copyright 2026 Anthony J. Arnovitz All Rights Reserved

Example 3

Ok, that is all well and good, but what if you don't want to toggle a pin but instead, you want to call a function in your code on a repeating basis on a specified time interval? This class supports that capability with a constructor that has a parameter for the time duration, the pin to toggle, and a third parameter that contains the address of the function you wish to be called each time when the time duration has passed, as follows:

```
ArnoTek_HeartBeat  Seconds90((long unsigned)90000, 0, &My90SecFunction);
```

As before, the first parameter to the constructor is a ***long unsigned*** value and in this case specifies a time duration value of 90,000 milliseconds, or 90 seconds. The second parameter specifies a pin value of 0 which indicates that no GPIO pin is to be toggled, and the third parameter is the address to the function that is to be called each time the duration passes. No code is needed in the ***setup*** function and the code is as follows.

```
// Instantiate the timer objects
ArnoTek_HeartBeat       HeartBeat;
ArnoTek_HeartBeat       Pin12_signal((long unsigned)60000, 12);
ArnoTek_HeartBeat       Seconds90((long unsigned)90000, 0, &My90SecFunction);


void loop
{
        HeartBeat.Toggle();         // heartbeat LED
        Pin12_signal.Toggle();      // Toggle pin 12 every minute
        Seconds90.Toggle();         // Call function specified earlier every 90 seconds
}


void My90SecFunction_1(ArnoTek_HeartBeat *myObject
                       , long unsigned currentTimer)
{
        // This code will run once every 90 seconds.  The first input parameter (myObject) is
        // a pointer to the HeartBeat object that called the callback function and the second
        // input parameter (currentTimer)  contains the current value of the millis() timer
        // when this function is called.
}
```

Copyright 2026 Anthony J. Arnovitz All Rights Reserved

Example 4

What about a case that is similar to what we already discussed, but now you want another function to run every 5 minutes in addition to the 90 second function? Basically, all we have to do is add an additional instantiation of the ***HeartBeat*** object and set its duration to 5 minutes and give it the address of the function that is to be processed every 5 minutes.

```
// Instantiate the timer objects
ArnoTek_HeartBeat   HeartBeat;
ArnoTek_HeartBeat   Pin12_signal((long unsigned)60000), 12);
ArnoTek_HeartBeat   Seconds90((long unsigned)90000), 0, & My90SecFunction);
ArnoTek_HeartBeat   Minutes5((long unsigned)300000) , 0, &My5MinFunction);


void loop
{
        HeartBeat.Toggle();         // heartbeat LED
        Pin12_signal.Toggle();      // Toggle pin 12 every minute
        Seconds90.Toggle();         // Call the function target every 90 seconds
        Minutes5.Toggle();          // Call the function target once every 5 minutes
}


void My90SecFunction(ArnoTek_HeartBeat *myObject
                                , long unsigned currentTimer)
{
        //  This code will run once every 90 seconds.  The first input parameter (myObject) is
        // a pointer to the HeartBeat object that invoked the callback function and the
        // second input parameter (currentTimer)  contains the current value of the
        // millis() timer when this function is called.
}

void My5MinFunction(ArnoTek_HeartBeat *myObject
                                , long unsigned currentTimer)
{
        // This code will run once every 5 minutes.  The first input parameter (myObject) is
        // a pointer to the HeartBeat object that invoked the callback function and
        // the second input parameter (currentTimer)  contains the current value
        // of the millis() timer when this function is called.
}
```

Copyright 2026 Anthony J. Arnovitz All Rights Reserved

Example 5

What about a case that is close to what we already discussed, but now you want a routine to run every 5 minutes but you only want the 90 second function to run only once? In the Seconds90 routine, it needs to disable itself so it won't be called again. That can be done by setting the ***Callback*** modifier and setting it to a ***nullptr*** value as follows. Note that the ***HeartBeat*** code will still be checking if there is a callback function every 90 seconds. To reduce this overhead, it is advisable to also set the timer delay to a very high value.

```
// Instantiate the timer objects
ArnoTek_HeartBeat   HeartBeat;
ArnoTek_HeartBeat   Pin12_signal((long unsigned)60000, 12);
ArnoTek_HeartBeat   Seconds90((long unsigned)90000, 0, &My90SecFunction);
ArnoTek_HeartBeat   Minutes5((long unsigned)300000, 0, &My5MinFunction);


void loop
{
        HeartBeat.Toggle();         // heartbeat LED
        Pin12_signal.Toggle();      // Toggle pin 12 every minute
        Seconds90.Toggle();         // Call the function target every 90 seconds
        Minutes5.Toggle();          // Call the function target once every 5 minutes
}
```

Copyright 2026 Anthony J. Arnovitz All Rights Reserved

```
void My90SecFunction(ArnoTek_HeartBeat *myObject
                                , long unsigned currentTimer)
{
        //  This code will run once at 90 seconds.  The first input parameter (myObject) is
        // a pointer to the HeartBeat object that called the callback function and the second
        // input parameter (currentTimer)  contains the current value of the millis() timer
        // when this function is called.

        myObject->CallBack(nullptr);      // Disable callback
        myObject->Delay(4294967000);   // Set the timer delay to a very high number
}


void My5MinFunction(ArnoTek_HeartBeat *myObject, long unsigned myCurrentTimer)
{
        // This code will run once every 5 minutes.  The first input parameter (myObject) is
        // a pointer to the HeartBeat object that called the callback function and the second
        // input parameter (currentTimer)  contains the current value of the millis() timer
        // when this function is called.
}
```

Copyright 2026 Anthony J. Arnovitz All Rights Reserved

Example 6

Besides the multiple constructors to support these capabilities, there is also a series of Init functions that are typically placed in the ***setup*** function. These are used instead of the constructors with parameters. Each of the desired heartbeat objects must be instantiated as before but this time with no parameters. They are used as follows:

```
// Instantiate the timer objects
ArnoTek_HeartBeat   HeartBeat;
ArnoTek_HeartBeat   Pin12_signal;
ArnoTek_HeartBeat   Seconds90;
ArnoTek_HeartBeat   Minutes5;
```

Each of the HeartBeat objects must then be initialized in the ***setup*** function.

```
void setup
{
        HeartBeat.Init((long unsigned)250);
        Pin12_signal.Init((long unsigned)60000, 12);
        Seconds90.Init((long unsigned)90000, 0, &My90SecFunction);
        Minutes5.Init((long unsigned)300000, 0, &My5MinFunction);
}
```

The ***loop*** function is exactly as before.

```
void loop
{
        HeartBeat.Toggle();         // heartbeat LED
        Pin12_signal.Toggle();      // Toggle pin 12 every minute
        Seconds90.Toggle();         // Call the function target every 90 seconds
        Minutes5.Toggle();          // Call the function target every 5 minutes
}
```

Copyright 2026 Anthony J. Arnovitz All Rights Reserved

```
void My90SecFunction (ArnoTek_HeartBeat *myObject
                                , long unsigned myCurrentTimer)
{
        // This code will run once at 90 seconds.  The first input parameter (myObject) is
        // a pointer to the HeartBeat object that called the callback function and the second
        // input parameter (currentTimer)  contains the current value of the millis() timer
        // when this function is called.

        // The invoking of the callback function is disable by setting its value to "nullptr".
        myObject->CallBack = nullptr;     // Disable callback
        myObject->Delay = 4294967000;  // Set the timer delay to a very high number
}
```

Note that the ***HeartBeat*** class also has modifiers to change the pin number where a value of 0 indicates not toggle any pins. (Note that there is no verification that the specified parameter is a valid pin number.)

Copyright 2026 Anthony J. Arnovitz All Rights Reserved

```
// This callback function sets up toggling pins the first time it is caller
// and then alternates which pin is toggled every other time it is called.
// This produces the effect of toggling one pin on then off at the timeout
// value and then it changes to a different pin which is toggled on and off
// and then it switches back to the initial pin.

void My5MinFunction (ArnoTek_HeartBeat *myObject, long unsigned myCurrentTimer)
{
        // This code will run once every 5 minutes.  The first input parameter (myObject) is
        // a pointer to the HeartBeat object that called the callback function and the second
        // input parameter (currentTimer)  contains the current value of the millis() timer
        // when this function is called.

        // This function will configure the 2 target pins as output and then swap back and
        // forth which pins are being toggled after each cycle.

        Serial.print("\tMy5MinFunction entered - current timer value: ");
        Serial.println(myCurrentTime);
        static int flagCount  = 0;
        static bool flag        = false;
        static bool firstTime = true;

        // If this is the first time this function is called, it needs to
        // mark the GPIO pins as output because the pins were not initialized
        // as output when the HeartBeat object was initialized since the pin
        // to be toggled was originally set to 0.
        if(firstTime)
        {
         pinMode(10, OUTPUT);
         pinMode(11, OUTPUT);
         firstTime = false;
        }

        // Swap which pins are being toggled after toggling the pin on then off
        if((flagCount % 2) == 0 )
        {
         if(flag) myObject->Pin(10);
         else myObject->Pin(11);
         flag = !flag;
        }
        flagCount++;
}
```

Copyright 2026 Anthony J. Arnovitz All Rights Reserved

Example 7

But why bother with using the Init functions when you can accomplish the same thing with the constructors? The Init functions also can provide an additional parameter for the initial **millis()** value. This enables all of the HeartBeat objects to use the same initial time value. The ***Toggle*** function also supports specifying the timer value. This provides the capability for all of the ***HeartBeat*** objects to use the same timer value for their processing. For example:

```
// Instantiate the timer objects
ArnoTek_HeartBeat   HeartBeat;
ArnoTek_HeartBeat   Pin12_signal;
ArnoTek_HeartBeat   Seconds90;
ArnoTek_HeartBeat   Minutes5;
```

Each of the ***HeartBeat*** objects must then be initialized in the setup function. Note that the ***Init*** functions need to be supplied with an initial ***millis*** timer value;

```
void setup
{
  // Get the current millisecond timer value for initializing all of the HeartBeat objects
  long unsigned initialTimer = millis();

  // Initialize all of the heartbeat objects using the identical millisecond  timer value
  HeartBeat.Init(initialTimer, (long unsigned)500);
  Pin12_signal.Init(initialTimer, (long unsigned)60000, 12);
  Seconds90.Init(initialTimer, (long unsigned)90000, 0, &My90SecFunction);
  Minutes5.Init(initialTimer, (long unsigned)300000, 0, &My5MinFunction);
}
```

The loop function and the callback functions are exactly as before but with one additional parameter.

```
void loop
{
  // Get the current millisecond timer value
  long unsigned currentTimer = millis();

  // Call each of the HeartBeat toggle functions using the identical
  // millisecond timer value
  HeartBeat.Toggle(currentTimer);     // heartbeat LED
  Pin12_signal.Toggle(currentTimer);  // Toggle pin 12 every minute
  Seconds90.Toggle(currentTimer);     // Call the function target every 90 seconds
  Minutes5.Toggle(currentTimer);      // Call the function target every 5 minutes
}
```

Copyright 2026 Anthony J. Arnovitz All Rights Reserved

```
void My90SecFunction(ArnoTek_HeartBeat *myObject, long unsigned myCurrentTimer)
{
        //  This code will run once at 90 seconds.  The first input parameter (myObject) is
        // a pointer to the HeartBeat object that called the callback function and the second
        // input parameter (currentTimer)  contains the current value of the millis() timer
        // when this function is called.

        myObject->CallBack = nullptr;     // Disable callback
        myObject->Delay = 4294967000;  // Set the timer delay to a very high number
}


void My5MinFunction _5(ArnoTek_HeartBeat *myObject
                                , long unsigned myCurrentTimer)
{
        //  This code will run once every 5 minutes.  The first input parameter (myObject) is
        // a pointer to the HeartBeat object that called the callback function and the second
        // input parameter (currentTimer)  contains the current value of the millis() timer
        // when this function is called.
}
```

Copyright 2026 Anthony J. Arnovitz All Rights Reserved

Example 8

The HeartBeat object contains three variables for the user to utilize however they see fit. These variables are not used by the HeartBeat object and they are all defined as unsigned long variables and may contain counters, identifiers, pointers, etc. They are named:

- UserSaveValue1
- UserSaveValue2
- UserSaveValue3

Note that in an earlier example, GPIO pin numbers were changed in the callback function. These pin numbers could have also been passed to the callback function using the user save values. The variables may also contain function pointers as shown in this example.

```
// Instantiate the timer objects
ArnoTek_HeartBeat   HeartBeat;

// NOTE: In this particular example, the same callback function is specified
//          for both the following HeartBeat objects.
ArnoTek_HeartBeat   Pin12_signal((long unsigned)2000, 12, &Callback);
ArnoTek_HeartBeat   Pin11_signal((long unsigned)5500, 11, &Callback);



void loop
{
        HeartBeat.Toggle();          // heartbeat LED
        Pin12_signal.Toggle();       // Toggle pin 12 every two seconds
        Pin11_signal.Toggle();       // Toggle pin 11 every five and half seconds
}
```

Copyright 2026 Anthony J. Arnovitz All Rights Reserved

```
//    *** User Defined Functions ***

// This function is called from multiple HeartBeat objects.  It in turn calls
// the function that was specified in the calling HeartBeat's variable
// "UserSaveValue1" and passes to it the value that was saved in the
// calling HeartBeat's "UserSaveValue2".

void Callback(ArnoTek_HeartBeat *myObject, long unsigned myCurrentTime)
{
 Serial.print("\n\nCallback entered for object: ");
 Serial.println((uintptr_t)myObject, HEX);


 // Define a function pointer and set it to the address of the function that is to be
 // called from the calling HeartBeat object's saved user save value

 void(*myNextFunction)(long unsigned) = myObject->UserSaveValue1();

 // Call the specified function using the function pointer and pass it the
 // value that was saved in the calling HeartBeat object's second
 // saved user save value

 myNextFunction(myObject->UserSaveValue2());

 Serial.print("Callback exited for object: ");
 Serial.println((uintptr_t)myObject, HEX);
}



// This function is called from the function that is specified in the HeartBeat object.

void Function1(long unsigned myParm)
{
 Serial.println("\tFunction 1 entered");

 Serial.print("\t\tThe parameter value is: ");
 Serial.println(myParm);

 Serial.println("\tFunction 1 exited ");
}
```

Copyright 2026 Anthony J. Arnovitz All Rights Reserved

```
// This function is called from the function that is specified in the HeartBeat object.

void Function2(long unsigned myParm)
{
 Serial.println("\tFunction 2 entered");

 Serial.print("\t\tThe parameter value is: ");
 Serial.println(myParm);

 Serial.println("\tFunction 2 exited ");
}
```

Copyright 2026 Anthony J. Arnovitz All Rights Reserved