Lessons: 00.Arduino-examples

01. Basics

Version: 0.0.5, Date: 2016-10-24

Contents

Bare Minimum

Bare Minimum Notes

- Intentionally left blank?
- Something must be missing?
- Cosa has default `setup()` and `loop()` functions.
- The sketch can skip one (or both).

Blink

```
#include “Cosa/OutputPin.hh”          // Explicit include header file

OutputPin led(Board::LED);           // Output Pin Instance for LED

void loop()
{
  led.write(1);                      // Turn LED on and off
  delay(1000);                       // With one second delay
  led.write(0);
  delay(1000);
}
```

Blink Notes

- Explicit include of components used in the sketch.
- Pins are symbols e.g. `Board::LED`.
- The `setup()` not needed.
- Pin initiated by the constructor.

Cont. Blink Notes

- The `OutputPin` class allows several methods of setting the pin:
  - `Pin.write(`*`value`*`)`
  - `Pin = `*`value`*
  - `Pin.on(), Pin.off()`
  - `Pin.set(), Pin.clear()`
  - `Pin.high(), Pin.low()`
  - `Pin.toggle()`
- `Select according to sketch/pin usage.`

Blink Assignment

```
#include “Cosa/OutputPin.hh”

OutputPin led(Board::LED);            // Output Pin Instance for LED

void loop()
{
  led = 1;                            // Turn LED on and off
  delay(1000);                        // With one second delay
  led = 0;
  delay(1000);
}
```

Blink On Off

```
#include “Cosa/OutputPin.hh”          // Explicit include header file

OutputPin led(Board::LED);           // Output Pin Instance for LED

void loop()
{
  led.on();                          // Turn LED on and off
  delay(1000);                       // With one second delay
  led.off();
  delay(1000);
}
```

Blink Set Clear

```
#include “Cosa/OutputPin.hh”          // Explicit include header file

OutputPin led(Board::LED);           // Output Pin Instance for LED

void loop()
{
  led.set();                         // Turn LED on and off
  delay(1000);                       // With one second delay
  led.clear();
  delay(1000);
}
```

Blink Toggle

```
#include “Cosa/OutputPin.hh”          // Explicit include header file

OutputPin led(Board::LED);           // Output Pin Instance for LED

void loop()
{
  led.toggle();                      // Change LED state
  delay(1000);                       // Every second
}
```

Alternative delay()

- The default `delay()` is implemented with *busy-wait*. More efficient is to use the Real-Time Timer (RTT) or the `Watchdog`.
- The RTT gives a hardware timer based clock with one milli-second tick. It also implements `micros()` and `millis()`, and allows low-power mode (Timer2).
- The `Watchdog` gives additional low-power but is less accurate and has a max resolution of 16 ms per tick.

Blink RTT Delay

```
#include “Cosa/OutputPin.hh”          // Explicit include header files
#include “Cosa/RTT.hh”

OutputPin led(Board::LED);           // Output Pin Instance for LED

void setup()
{
  RTT::begin();                      // Start Real-Time Timer
}

void loop()
{
  led.on();                          // Turn LED on
  delay(1000);                       // Delay 1000 ms
  led.off();                         // Turn LED off
  delay(1000);                       // Delay 1000 ms
}
```

Blink Watchdog Delay

```
#include “Cosa/OutputPin.hh”          // Explicit include header files
#include “Cosa/Watchdog.hh”

OutputPin led(Board::LED);           // Output Pin Instance for LED

void setup()
{
  Watchdog::begin();                 // Start Watchdog
}

void loop()
{
  led.on();                          // Turn LED on
  delay(1000);                       // Delay 1000 ms
  led.off();                         // Turn LED off
  delay(1000);                       // Delay 1000 ms
}
```

Digital Read Serial

```
#include “Cosa/InputPin.hh”          // Explicit include header files
#include “Cosa/IOStream.hh”
#include “Cosa/UART.hh”

InputPin button(Board::D2);          // Input Pin Instance for D2
IOStream ios(&uart);                 // IOStream Instance bound to UART

void setup()
{
  uart.begin(9600);                  // Start UART and use 9600 baud
}

void loop()
{
  ios << button << endl;             // Print digital pin reading
  delay(100);                        // every 100 ms
}
```

Digital Read Serial Notes

- Component initialization in `setup()` i.e. the `uart` is initiated to 9600 baudrate and default format (`8-N-2`).
- `IOStream` instance `ios` delegates to UART which implements the `IOStream::Device` interface.
- `IOStream` print with `operator<<` and `endl`.
- The Pin value is read when the identifier is used in an expression (*value-of-pin*).

Cont. Digital Read Serial Notes

- The `InputPin` class allows several methods of accessing the pin:
  - *value =* `Pin.read()`
  - *value =* `Pin`
  - `if (Pin) .., while (Pin) ...`
  - `Pin.is_on(), Pin.is_off()`
  - `Pin.is_set(), Pin.is_clear()`
  - `Pin.is_high(), Pin.is_low()`
- `Select according to sketch/pin usage.`

Fade

```
#include "Cosa/PWMPin.hh"              // Explicit include header file

PWMPin led(Board::PWM3);               // PWM Pin Instance on PWM3/D9
int brightness = 0;                    // Current brightness level
int fadeAmount = 5;                    // Brightness adjust amount

void setup()
{
  led.begin();                         // Start the PWM pin
}

void loop()
{
  led = brightness;
  brightness += fadeAmount;
  if (brightness <= 0 || brightness >= 255)
    fadeAmount = -fadeAmount;
  delay(10);
}
```

Analog Read Serial

```
#include “Cosa/AnalogPin.hh”          // Explicit include header files
#include “Cosa/IOStream.hh”
#include “Cosa/UART.hh”

AnalogPin sensor(Board::A0);          // Analog Pin Instance for A0
IOStream ios(&uart);                  // IOStream Instance bound to UART

void setup()
{
  uart.begin(9600);                   // Start UART and use 9600 baud
  AnalogPin::powerup();               // Start ADC module
}

void loop()
{
  ios << sensor << endl;              // Print analog pin reading
  delay(100);                         // Every 100 ms
}
```

Analog Read Serial Notes

- The ADC hardware module is not automatically started by Cosa.
- A sketch that used `AnalogPin` needs to powerup ADC.

Voltage Read Serial

```
#include “Cosa/AnalogPin.hh”          // Explicit include header files
#include “Cosa/IOStream.hh”
#include “Cosa/UART.hh”

AnalogPin sensor(Board::A0);          // Analog Pin Instance for A0
IOStream ios(&uart);                  // IOStream Instance bound to UART

void setup()
{
  uart.begin(9600);                   // Start UART and use 9600 baud
  AnalogPin::powerup();               // Start ADC module
}

void loop()                           // Convert to voltage and print
{
  float voltage = sensor * (5.0 / 1023.0);
  ios << voltage << endl;
  delay(100);
}
```

License: LGPL-2.1