Arduino code examples used for the unit ‘SMB–Science Magic Box’

Program that converts Temperature into Light Intensity

```
int RedLedPin = 6;   // Connect Led on Digital Pin 6 ; Resistance 1 KOhm
int LEDbrightness;
void setup()
{
   Serial.begin(9600);//Set Baud Rate to 9600 bps
   pinMode(LED_BUILTIN, OUTPUT);
   pinMode(RedLedPin, OUTPUT);
}

unsigned int intdelay=1000;

void loop()
{
   uint16_t val;
   double dat;
   uint16_t InAnalogLight;



   val=analogRead(A0);//Connect LM35 on Analog 0
   dat = (double) val * (3.3/10.24);  // Vref su Due 3.3 V
   Serial.print("Temp:"); //Display the temperature on Serial monitor
   Serial.print(dat);
   Serial.println("°C");
   Serial.println(val);
   delay(intdelay);
   LEDbrightness = map(val, 60, 85, 0, 255);
```

```
  analogWrite(RedLedPin, LEDbrightness); //Analog Write to Digital PWM pin
  delay(200);

  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(intdelay);                   // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW

}
```

Program that reads the pressure force sensor on analogue pin A0, transduces the signal to a PWM-modulated Buzzer for an audio signal and into a light signal on a LED, controlled in intensity thanks to the PWM feature on digital out.

```
int buzzPin =  8;    //Connect Buzzer on Digital Pin3
int RedLedPin = 6;   // Connect Led on Digital Pin 6 ; Resistance 1 KOhm

int LEDbrightness;

#define NOTE_B3 493
#define NOTE_C4 523
#define NOTE_D4 587
#define NOTE_E4 659
#define NOTE_F4 698
#define NOTE_G4 783
#define NOTE_A4 880
#define NOTE_B4 987
#define NOTE_C5 1046
#define NOTE_D5 1174
#define NOTE_E5 1318
#define NOTE_F5 1318
```

```
#define NOTE_DURATION 1000

// note durations: 4 = quarter note, 8 = eighth note, etc.:
int noteDurations[] = {
  2, 2, 2, 2,2,2,2,2 };

uint16_t prova;

int fsrPin = 0;     // the FSR and 10K pulldown are connected to a0
int fsrReading;     // the analog reading from the FSR resistor divider

void setup(void) {
  Serial.begin(9600);
  pinMode(LED_BUILTIN, OUTPUT);
  pinMode(buzzPin, OUTPUT);
  pinMode(RedLedPin, OUTPUT);



}

void loop(void) {
  fsrReading = analogRead(fsrPin);
  delay(10);
  fsrReading = analogRead(fsrPin);

  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)

  LEDbrightness = map(fsrReading, 0, 1023, 0, 255);
  // LED gets brighter the harder you press
```

```
analogWrite(RedLedPin, LEDbrightness); //Analog Write to Digital PWM pin



prova= pulseIn(1,HIGH);

Serial.print("Analog reading = ");
Serial.print(fsrReading);     // the raw analog reading


if (fsrReading == 0) {
  Serial.println(" - No pressure");
} else if (fsrReading < 50) {
  Serial.println(" - Light touch");
  tone(8, NOTE_B3,NOTE_DURATION);
} else if (fsrReading < 100) {
  Serial.println(" - Light touch");
  tone(8, NOTE_C4,NOTE_DURATION);
} else if (fsrReading < 200) {
  Serial.println(" - Light touch");
  tone(8, NOTE_D4,NOTE_DURATION);
} else if (fsrReading < 300) {
  Serial.println(" - Light touch");
  tone(8, NOTE_E4, NOTE_DURATION);
} else if (fsrReading < 400) {
  tone(8, NOTE_F4,NOTE_DURATION);
  Serial.println(" - Light squeeze");
} else if (fsrReading < 500) {
  tone(8, NOTE_G4,NOTE_DURATION);
  Serial.println(" - Medium squeeze");
```

```
  } else if (fsrReading < 600) {
    tone(8, NOTE_A4,NOTE_DURATION);
    Serial.println(" - Medium squeeze");
  } else if (fsrReading < 700) {
    tone(8, NOTE_B4,NOTE_DURATION);
    Serial.println(" - Medium squeeze");
  } else if (fsrReading < 800) {
    tone(8, NOTE_C5,NOTE_DURATION);
    Serial.println(" - Medium squeeze");
  } else if (fsrReading < 900) {
    tone(8, NOTE_D5,NOTE_DURATION);
    Serial.println(" - Medium squeeze");
  } else if (fsrReading < 1000) {
    tone(8, NOTE_E5,NOTE_DURATION);
    Serial.println(" - Medium squeeze");
  } else {
    tone(8, NOTE_F5,NOTE_DURATION);
    Serial.println(" - Big squeeze");
  }
 // delay(300);
 // noTone(8);

  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
//  delay(200);                       // wait for a second


}


/*
```

```
Tone generator
v1  use timer, and toggle any digital pin in ISR
   funky duration from arduino version
   TODO use FindMckDivisor?
   timer selected will preclude using associated pins for PWM etc.
    could also do timer/pwm hardware toggle where caller controls duration
*/


// timers TC0 TC1 TC2   channels 0-2 ids 0-2  3-5  6-8     AB 0 1
// use TC1 channel 0
#define TONE_TIMER TC1
#define TONE_CHNL 0
#define TONE_IRQ TC3_IRQn

// TIMER_CLOCK4   84MHz/128 with 16 bit counter give 10 Hz to 656KHz
//  piano 27Hz to 4KHz

static uint8_t pinEnabled[PINS_COUNT];
static uint8_t TCChanEnabled = 0;
static boolean pin_state = false ;
static Tc *chTC = TONE_TIMER;
static uint32_t chNo = TONE_CHNL;

volatile static int32_t toggle_count;
static uint32_t tone_pin;

// frequency (in hertz) and duration (in milliseconds).

void tone(uint32_t ulPin, uint32_t frequency, int32_t duration)
```

```
{
    const uint32_t rc = VARIANT_MCK / 256 / frequency;
    tone_pin = ulPin;
    toggle_count = 0;  // strange  wipe out previous duration
    if (duration > 0 ) toggle_count = 2 * frequency * duration / 1000;
    else toggle_count = -1;

    if (!TCChanEnabled) {
      pmc_set_writeprotect(false);
      pmc_enable_periph_clk((uint32_t)TONE_IRQ);
      TC_Configure(chTC, chNo,
        TC_CMR_TCCLKS_TIMER_CLOCK4 |
        TC_CMR_WAVE |         // Waveform mode
        TC_CMR_WAVSEL_UP_RC ); // Counter running up and reset when equals to
RC

      chTC->TC_CHANNEL[chNo].TC_IER=TC_IER_CPCS;  // RC compare interrupt
      chTC->TC_CHANNEL[chNo].TC_IDR=~TC_IER_CPCS;
      NVIC_EnableIRQ(TONE_IRQ);
                        TCChanEnabled = 1;
    }
    if (!pinEnabled[ulPin]) {
      pinMode(ulPin, OUTPUT);
      pinEnabled[ulPin] = 1;
    }
    TC_Stop(chTC, chNo);
                TC_SetRC(chTC, chNo, rc);    // set frequency
    TC_Start(chTC, chNo);
}
```

```
void noTone(uint32_t ulPin)
{
  TC_Stop(chTC, chNo);  // stop timer
  digitalWrite(ulPin,LOW);  // no signal on pin
}

// timer ISR  TC1 ch 0
void TC3_Handler ( void ) {
  TC_GetStatus(TC1, 0);
  if (toggle_count != 0){
    // toggle pin  TODO  better
    digitalWrite(tone_pin,pin_state= !pin_state);
    if (toggle_count > 0) toggle_count--;
  } else {
    noTone(tone_pin);
  }
}
```

The following program converts an input environment noise signal into a light signal, using LEDs with different colours, turning on LEDs according to noise intensity, being RED, YELLOW and GREEN used in this order for increasing noise level.

```
int PIN_LED_GREEN = 4;   // Connect Led on Digital Pin 6 ; Resistance 1 KOhm
int PIN_LED_RED   = 6;   // Connect Led on Digital Pin 6 ; Resistance 1 KOhm
int PIN_LED_YELLOW   = 7;   // Connect Led on Digital Pin 6 ; Resistance 1 KOhm

void setup()
{
   Serial.begin(9600);//Set Baud Rate to 9600 bps
```

```
    pinMode(LED_BUILTIN, OUTPUT);
    pinMode(PIN_LED_RED, OUTPUT);
    pinMode(PIN_LED_GREEN, OUTPUT);
    pinMode(PIN_LED_YELLOW, OUTPUT);
}

unsigned int intdelay=200;

void loop()
{
    uint16_t val;
    double dat;
    uint16_t InAnalogNoise;

   // The trick when using multiple analog sensors is to read them twice,
   // with a small delay after each read (10ms is good), then discard the first reading.
   // This is because the ADC multiplexer needs switching time and the voltage needs time to stabilize after switching..
   // Basically the first analogRead call causes the multiplexer to switch, the delay gives the voltage time to stabilize,
   // then your second read should be much more accurate with less jitter.


   InAnalogNoise=analogRead(0);   //connect mic sensor to Analog 0
   delay(10);
   InAnalogNoise=analogRead(0);   //connect mic sensor to Analog 0
   Serial.print("Noise : ");
   Serial.println(InAnalogNoise,DEC);//print the sound value to serial
   Serial.println();
   delay(intdelay);
```

```
 if (InAnalogNoise>100)
  {
  digitalWrite(PIN_LED_RED, HIGH);
  if (InAnalogNoise>400)
     {
      digitalWrite(PIN_LED_YELLOW, HIGH);
      if (InAnalogNoise>700)
        {
         digitalWrite(PIN_LED_GREEN, HIGH);
        }
      else
        {
         digitalWrite(PIN_LED_GREEN, LOW);
        }
    }
  else
     {
      digitalWrite(PIN_LED_YELLOW, LOW);
     }
  }
 else
  {
   digitalWrite(PIN_LED_RED, LOW);
   digitalWrite(PIN_LED_YELLOW, LOW);
   digitalWrite(PIN_LED_GREEN, LOW);
  }
delay(80);
```

```
  digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(intdelay);                   // wait for a second
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW

}
```

The following program reads Ambient Analog Light sensor from A0 Analog Pin used as INPUT. Transduces the signal into an acoustic signal on Buzzer connected on Digital PIN 8 frequency modulated thanks to its PWM feature. Transduces the signal into a light intensity signal on digital pin 6 connected RED LED, written as ANALOG OUTPUT USING DIGITAL PIN PWM feature: more light from LED, when ambient luminosity decreases. Issues a sound with increasing frequency when light intensity increases (see the above code for sound emission).

```
 int PIN_LED_RED   = 6;   // Connect Led on Digital Pin 6 ; Resistance 1 KOhm

int buzzPin =  8;    //Connect Buzzer on Digital Pin3
int RedLedPin = 6;   // Connect Led on Digital Pin 6 ; Resistance 1 KOhm

int LEDbrightness;

// Musical notes frequencies mapping
#define NOTE_B3 493
#define NOTE_C4 523
#define NOTE_D4 587
#define NOTE_E4 659
#define NOTE_F4 698
#define NOTE_G4 783
#define NOTE_A4 880
#define NOTE_B4 987
```

```
#define NOTE_C5 1046
#define NOTE_D5 1174
#define NOTE_E5 1318
#define NOTE_F5 1318



// note durations: 4 = quarter note, 8 = eighth note, etc.:
int noteDurations[] = {
  2, 2, 2, 2,2,2,2,2 };

uint16_t prova;

int fsrPin = 0;     // the FSR and 10K pulldown are connected to a0
int fsrReading;     // the analog reading from the FSR resistor divider



void setup(void) {
  Serial.begin(9600);
  pinMode(LED_BUILTIN, OUTPUT);
  pinMode(buzzPin, OUTPUT);
  pinMode(RedLedPin, OUTPUT);



}

void loop(void) {
  fsrReading = analogRead(fsrPin);
  delay(10);
  fsrReading = analogRead(fsrPin);
```

```
prova= pulseIn(1,HIGH);

Serial.print("Analog reading = ");
Serial.println(fsrReading);     // the raw analog reading


LEDbrightness = map(fsrReading, 0, 1023, 255, 0);
// LED gets brighter the less luminosity you get from ambient
analogWrite(RedLedPin, LEDbrightness); //Analog Write to Digital PWM pin


Serial.print("Luce = ");
Serial.println(LEDbrightness);

 digitalWrite(LED_BUILTIN, HIGH);   // turn the LED on (HIGH is the voltage level)
 delay(200);                        // wait for 200 msec


if (fsrReading <= 100) {
} else if (fsrReading < 200) {
  tone(8, NOTE_B3,500);
} else if (fsrReading < 300) {
  tone(8, NOTE_C4,500);
} else if (fsrReading < 400) {
  tone(8, NOTE_D4,500);
} else if (fsrReading < 500) {
  tone(8, NOTE_E4,500);
} else if (fsrReading < 600) {
  tone(8, NOTE_F4,500);
```

```
  } else if (fsrReading < 700) {
    tone(8, NOTE_G4,500);
  } else if (fsrReading < 800) {
    tone(8, NOTE_A4,500);
  } else if (fsrReading < 900) {
    tone(8, NOTE_B4,500);
  } else if (fsrReading < 1000) {
    tone(8, NOTE_C5,500);
  }  else {
    tone(8, NOTE_D5,500);
  }

  delay(300);
  noTone(8);
  digitalWrite(LED_BUILTIN, LOW);    // turn the LED off by making the voltage LOW
  delay(200);                        // wait for a second
}
```