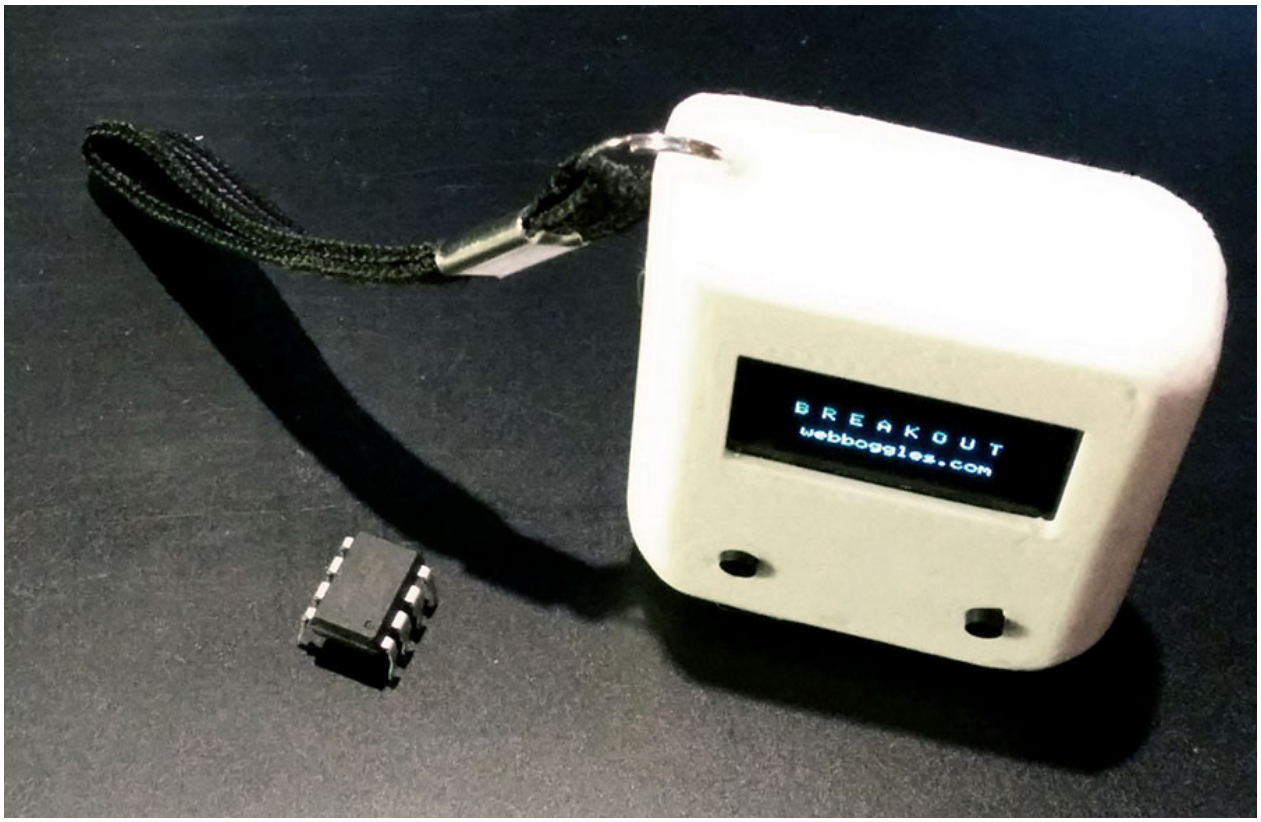
Attiny85 Breakout Keychain Game

So, what can you do with Attiny's 5 i/o pins?

UPDATE: New game, "UFO Escape" side scroller



I saw this great Attiny OLED project on http://tinusaur.wordpress.com/ and decided to try out the screen (It will be very handy as a mode display for a remote for my http://orb.photo/ project)

So I set out to make myself familiar with the screen and make something fun in the process. I made a remix of the classic breakout game, here is a video of the gameplay:

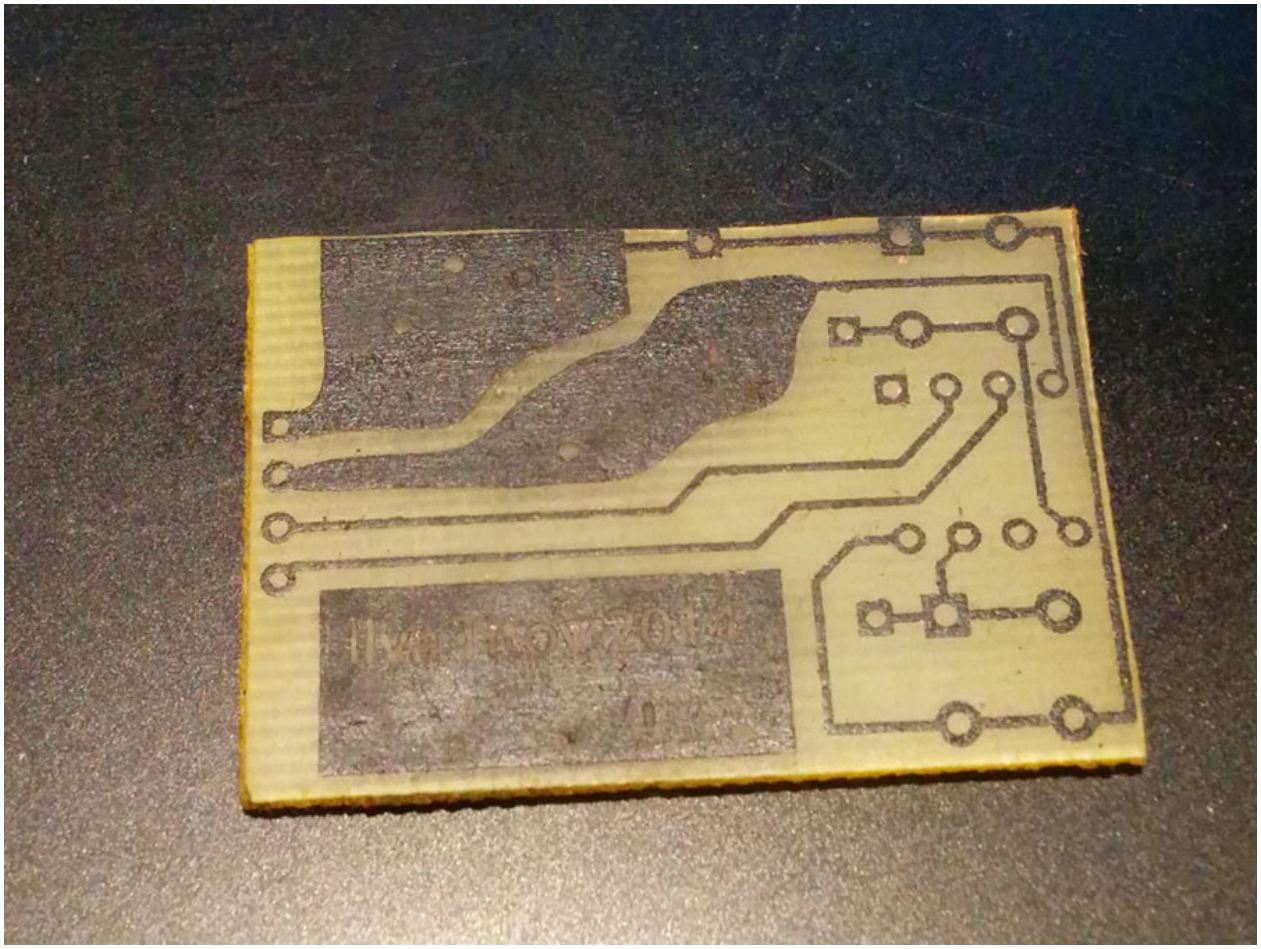
Breakout Keychain Game with Attiny85 and SSD1306 OLED Screen

Parts required:

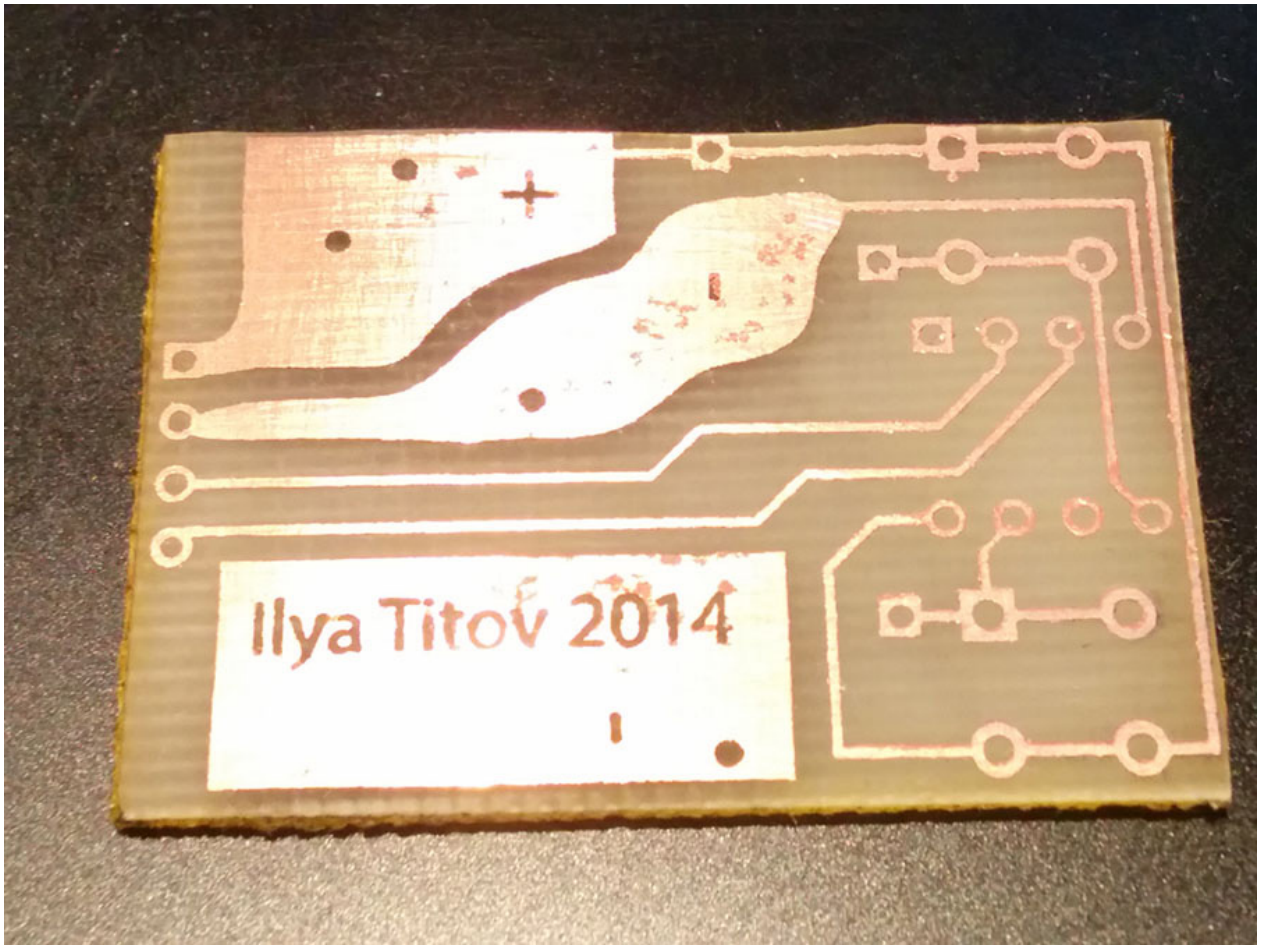
- Attiny85 + dip8 socket
- SSD1306 OLED screen
- 2x push buttons
- 2x resistors (10kOhm optimal)
- 3x4cm copper clad board
- Piezo speaker
- 3V 2032 coin cell battery
- Paper clip

So first of all you want to create the PCB, to do this just design the layout (printable pcb pdf)

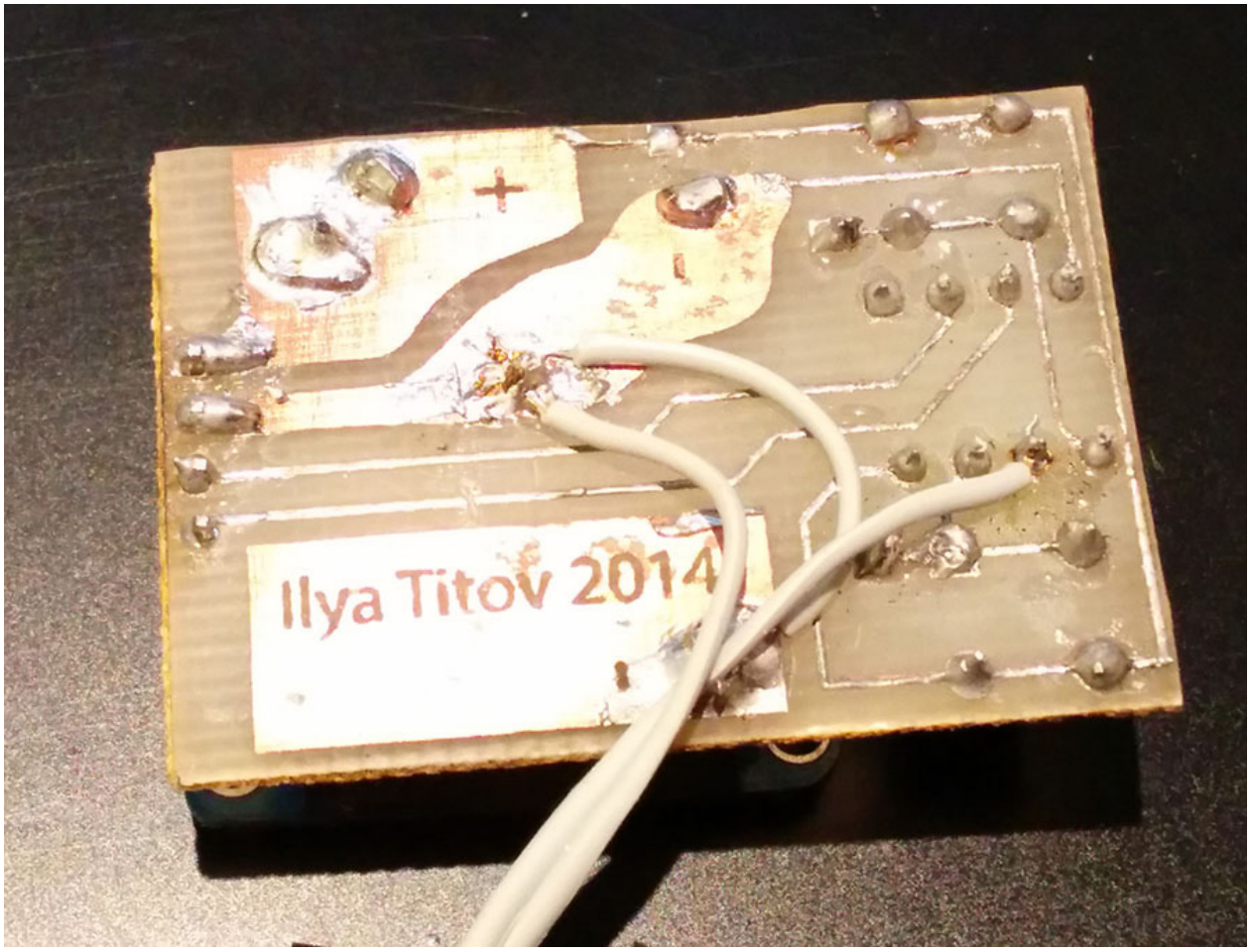
Print the design on a page from a glossy magazine with a laser printer and use an iron or a laminator to transfer the design to the copper. Put the PCB into ferric chloride until the copper is etched away



Clean off the toner with acetone:



Drill the holes and solder the parts. The speaker is soldered to PB1 pin on the attiny and ground. Connect the large ground pads with a bit of wire. Bend and solder the paper clip to keep the battery in place.



3D print the case from Thingiverse

Program the attiny85 with the code and enjoy:

```
/* 2014
 * Breakout game by Ilya Titov. Find building instructions on
http://webboggles.com/
 * The code that does not fall under the licenses of sources listed below can be
used non commercially with attribution.
 *
 * If you have problems uploading this sketch, this is probably due to sketch
size - you need to update ld.exe in arduino\hardware\tools\avr\avr\bin
 * https://github.com/TCWORLD/ATTinyCore/tree/master/PCREL%20Patch%20for%20GCC
 *
 * This sketch is using the screen control and font functions written by Neven
Boyanov for the http://tinusaur.wordpress.com/ project
 * Source code and font files available at:
https://bitbucket.org/tinusaur/ssd1306xled
 *
 * Sleep code is based on this blog post by Matthew Little:
 * http://www.re-innovation.co.uk/web12/index.php/en/blog-75/306-sleep-modes-on-
attiny85
*/
#include <EEPROM.h>
#include "font6x8.h"
#include <avr/sleep.h>
#include <avr/interrupt.h> // needed for the additional interrupt

volatile byte player = 0; //0 to 128-platformWidth  - this is the position of
the bounce platform
byte platformWidth = 16;
byte ballx = 62; // coordinate of the ball
byte bally = 50; // coordinate of the ball
int vdir = -1; // vertical direction and step  distance
int hdir = -1; // horizontal direction and step distance
long lastFrame = 0; // time since the screen was updated last
boolean row1[16]; // on-off array of blocks
boolean row2[16];
boolean row3[16];
int score = 0; // score - counts the number of blocks hit and resets the array
above when devisible by 48(total blocks)
ISR(PCINT0_vect){ // PB0 pin button interrupt
   if (player >0){player--;}
   return;
}
void playerInc(){ // PB2 pin button interrupt
  if (player <128-platformWidth){player++;}
}

void setup() {
  resetGame();
  DDRB = 0b00000010;    // set PB1 as output (for the speaker)
  PCMSK = 0b00000001;   // pin change mask: listen to portb bit 1
  GIMSK |= 0b00100000;  // enable PCINT interrupt
  sei();                          // enable all interrupts
  attachInterrupt(0,playerInc,CHANGE);
  lastFrame = millis();
```

```
}
void loop() {
  delay(40);
      noInterrupts();
      ssd1306_init();
      ssd1306_fillscreen(0x00);
      ssd1306_char_f6x8(16, 4, "B R E A K O U T");
      ssd1306_char_f6x8(20, 6, "webboggles.com");
      beep(200,600);          beep(300,200);          beep(400,300);
      delay(2000);
        while (1==1) {
              // continue moving after the interrupt
              if (digitalRead(2)==1){if (player <128-platformWidth){player++;}
if (player <128-platformWidth){player++;} if (player <128-platformWidth)
{player++;}}               if (digitalRead(0)==1){if (player >0){player--;} if
(player >0){player--;} if (player >0){player--;}}

              // bounce off the sides of the screen
              if ((bally+vdir<54&&vdir==1)||(bally-vdir>1&&vdir==-1))
{bally+=vdir;}else {vdir = vdir*-1;}
              if ((ballx+hdir<127&&hdir==1)||(ballx-hdir>1&&hdir==-1))
{ballx+=hdir;}else {hdir = hdir*-1;}

              // frame actions
              if (lastFrame+10<millis()){
if(bally>10&&bally+vdir>=54&&(ballx<player||ballx>player+platformWidth)){ //
game over if the ball misses the platform
                  int topScore = EEPROM.read(0);
                  topScore = topScore << 8;                    topScore =
topScore |  EEPROM.read(1);                                      if
(score>topScore){topScore = score; EEPROM.write(1,topScore & 0xFF);
EEPROM.write(0,(topScore>>8) & 0xFF); }
                  ssd1306_fillscreen(0x00);
                  ssd1306_char_f6x8(32, 3, "Game Over");
                  ssd1306_char_f6x8(32, 5, "score:");
                  char temp[4] = {0,0,0,0};
                  itoa(score,temp,10);
                  ssd1306_char_f6x8(70, 5, temp);
                  ssd1306_char_f6x8(32, 6, "top score:");
                  itoa(topScore,temp,10);
                  ssd1306_char_f6x8(90, 6, temp);
                  for (int i = 0; i<1000; i++){
                    beep(1,random(0,i*2));
                  }
                  delay(1000);
                  system_sleep();
                  resetGame();
                }else if
(ballx<player+platformWidth/2&&bally>10&&bally+vdir>=54){ // if the ball hits
left of the platform bounce left
                  hdir=-1; beep(20,600);
                }else if
(ballx>player+platformWidth/2&&bally>10&&bally+vdir>=54){  // if the ball hits
right of the platform bounce right
```

```
                  hdir=1; beep(20,600);
                }else if (bally+vdir>=54){
                  hdir=1; beep(20,600);
                }

                collisionCheck: // go back to here if a collision was detected
to prevent flying through a rigid
                if (floor((bally+vdir)/8)==2){
                  if (row3[ballx/8]==1){row3[ballx/8]=0; score++;
                      collision(); goto collisionCheck; // check collision for
the new direction to prevent flying through a rigid
                  }
                }else if (floor((bally+vdir)/8)==1){
                  if (row2[ballx/8]==1){row2[ballx/8]=0; score++;
                      collision(); goto collisionCheck;
                  }
                }else if (floor((bally+vdir)/8)==0){
                  if (row1[ballx/8]==1){row1[ballx/8]=0; score++;
                      collision(); goto collisionCheck;
                  }
                }

                // reset blocks if all have been hit
                if (score%48==0){
                  for (byte i =0; i<16;i++){
                   row1[i]=1; row2[i]=1; row3[i]=1;
                  }
                }
              }

              // update whats on the screen
                  noInterrupts();

                  // blocks
                  ssd1306_setpos(0,0);
                  ssd1306_send_data_start();
                  for (int bl = 0; bl <16; bl++){
                    if(row1[bl]==1){
                      sendBlock(1);
                    }else {
                      sendBlock(0);
                    }
                   }
                  ssd1306_send_data_stop();
                  ssd1306_setpos(0,1);
                  ssd1306_send_data_start();
                  for (int bl = 0; bl <16; bl++){
                    if(row2[bl]==1){
                      sendBlock(1);
                    }else {
                      sendBlock(0);
                    }
                   }
                  ssd1306_send_data_stop();
```

```
      ssd1306_setpos(0,2);
      ssd1306_send_data_start();
      for (int bl = 0; bl <16; bl++){
        if(row3[bl]==1){
          sendBlock(1);
        }else {
          sendBlock(0);
        }
       }
      ssd1306_send_data_stop();

      // clear area below the blocks
      ssd1306_setpos(0,3);
      ssd1306_send_data_start();
      for (byte i =0; i<128; i++){
         ssd1306_send_byte(B00000000);
      }
      ssd1306_send_data_stop();
      ssd1306_setpos(0,4);
      ssd1306_send_data_start();
      for (byte i =0; i<128; i++){
         ssd1306_send_byte(B00000000);
      }
      ssd1306_send_data_stop();
      ssd1306_setpos(0,5);
      ssd1306_send_data_start();
      for (byte i =0; i<128; i++){
         ssd1306_send_byte(B00000000);
      }
      ssd1306_send_data_stop();
      ssd1306_setpos(0,6);
      ssd1306_send_data_start();
      for (byte i =0; i<128; i++){
         ssd1306_send_byte(B00000000);
      }
      ssd1306_send_data_stop();
      ssd1306_setpos(0,7);
      ssd1306_send_data_start();
      for (byte i =0; i<128; i++){
         ssd1306_send_byte(B00000000);
      }
      ssd1306_send_data_stop();

      // draw ball
      ssd1306_setpos(ballx,bally/8);
      uint8_t temp = B00000001;
      ssd1306_send_data_start();
      temp = temp << bally%8+1;
      ssd1306_send_byte(temp);
      ssd1306_send_data_stop();

      drawPlatform();
      interrupts();
    //
```

```
        }
}
void resetGame(){
  ssd1306_char_f6x8(16, 4, "B R E A K O U T");
  ssd1306_char_f6x8(20, 6, "webboggles.com");
  beep(200,600);          beep(300,200);          beep(400,300);
  delay(2000);
  for (byte i =0; i<16;i++){ // reset blocks
   row1[i]=1; row2[i]=1; row3[i]=1;
  }
  platformWidth = 16;
  ballx = 64;
  bally = 50;
  hdir = -1;
  vdir = -1;
  score = 0;
  player = random(0,128-platformWidth);
  ballx = player+platformWidth/2;
}

void collision(){ // the collsision check is actually done befor this is called,
this code works out where the ball will bounce
  if ((bally+vdir)%8==7&&(ballx+hdir)%8==7){ // bottom right corner
      if (vdir==1){hdir=1;}else if(vdir==-1&&hdir==1){vdir=1;}else
{hdir=1;vdir=1;}
    }else if ((bally+vdir)%8==7&&(ballx+hdir)%8==0){ // bottom left corner
      if (vdir==1){hdir=-1;}else if(vdir==-1&&hdir==-1){vdir=1;}else
{hdir=-1;vdir=1;}
    }else if ((bally+vdir)%8==0&&(ballx+hdir)%8==0){ // top left corner
      if (vdir==-1){hdir=-1;}else if(vdir==1&&hdir==-1){vdir=-1;}else
{hdir=-1;vdir=-1;}
    }else if ((bally+vdir)%8==0&&(ballx+hdir)%8==7){ // top right corner
      if (vdir==-1){hdir=1;}else if(vdir==1&&hdir==1){vdir=-1;}else
{hdir=1;vdir=-1;}
    }else if ((bally+vdir)%8==7){ // bottom side
      vdir = 1;
    }else if ((bally+vdir)%8==0){ // top side
      vdir = -1;
    }else if ((ballx+hdir)%8==7){ // right side
      hdir = 1;
    }else if ((ballx+hdir)%8==0){ // left side
      hdir = -1;
    }else {
      hdir = hdir*-1; vdir = vdir*-1;
  }

  beep(30,300);

}
void drawPlatform(){
 noInterrupts();
 ssd1306_setpos(player,7);
 ssd1306_send_data_start();
```

```
 for (byte pw = 1; pw <platformWidth; pw++){ssd1306_send_byte(B00000011);}
 ssd1306_send_data_stop();
 interrupts();
}
void sendBlock(boolean fill){
  if (fill==1){
   ssd1306_send_byte(B00000000);
   ssd1306_send_byte(B01111110);
   ssd1306_send_byte(B01111110);
   ssd1306_send_byte(B01111110);
   ssd1306_send_byte(B01111110);
   ssd1306_send_byte(B01111110);
   ssd1306_send_byte(B01111110);
   ssd1306_send_byte(B00000000);
  }else {
   ssd1306_send_byte(B00000000);
   ssd1306_send_byte(B00000000);
   ssd1306_send_byte(B00000000);
   ssd1306_send_byte(B00000000);
   ssd1306_send_byte(B00000000);
   ssd1306_send_byte(B00000000);
   ssd1306_send_byte(B00000000);
   ssd1306_send_byte(B00000000);
  }
}
void beep(int bCount,int bDelay){
  for (int i = 0; i<=bCount; i++){digitalWrite(1,HIGH);for(int i2=0; i2<bDelay;
i2++){__asm__("nop\n\t");}digitalWrite(1,LOW);for(int i2=0; i2<bDelay; i2++)
{__asm__("nop\n\t");}}
}
#define DIGITAL_WRITE_HIGH(PORT) PORTB |= (1 << PORT)
#define DIGITAL_WRITE_LOW(PORT) PORTB &= ~(1 << PORT)

// Some code based on "IIC_wtihout_ACK" by http://www.14blog.com/archives/1358
#ifndef SSD1306XLED_H
#define SSD1306XLED_H
// ---------------------        // Vcc, Pin 1 on SSD1306 Board
// ---------------------        // GND, Pin 2 on SSD1306 Board
#ifndef SSD1306_SCL
#define SSD1306_SCL             PB3     // SCL, Pin 3 on SSD1306 Board
#endif
#ifndef SSD1306_SDA
#define SSD1306_SDA             PB4     // SDA, Pin 4 on SSD1306 Board
#endif
#ifndef SSD1306_SA
#define SSD1306_SA              0x78    // Slave address
#endif
// ---------------------------------------------------------------------------
void ssd1306_init(void);
void ssd1306_xfer_start(void);
void ssd1306_xfer_stop(void);
void ssd1306_send_byte(uint8_t byte);
void ssd1306_send_command(uint8_t command);
void ssd1306_send_data_start(void);
```

```
void ssd1306_send_data_stop(void);
void ssd1306_setpos(uint8_t x, uint8_t y);
void ssd1306_fillscreen(uint8_t fill_Data);
void ssd1306_char_f6x8(uint8_t x, uint8_t y, const char ch[]);
//void ssd1306_char_f8x16(uint8_t x, uint8_t y,const char ch[]);
//void ssd1306_char_f16x16(uint8_t x, uint8_t y, uint8_t N);
void ssd1306_draw_bmp(uint8_t x0, uint8_t y0, uint8_t x1, uint8_t y1, uint8_t
bitmap[]);
// ---------------------------------------------------------------------------
#endif
void ssd1306_init(void){
        DDRB |= (1 << SSD1306_SDA);     // Set port as output
        DDRB |= (1 << SSD1306_SCL);     // Set port as output

        ssd1306_send_command(0xAE); // display off
        ssd1306_send_command(0x00); // Set Memory Addressing Mode
        ssd1306_send_command(0x10); // 00,Horizontal Addressing Mode;01,Vertical
Addressing Mode;10,Page Addressing Mode (RESET);11,Invalid
        ssd1306_send_command(0x40); // Set Page Start Address for Page
Addressing Mode,0-7
        ssd1306_send_command(0x81); // Set COM Output Scan Direction
        ssd1306_send_command(0xCF); // ---set low column address
        ssd1306_send_command(0xA1); // ---set high column address
        ssd1306_send_command(0xC8); // --set start line address
        ssd1306_send_command(0xA6); // --set contrast control register
        ssd1306_send_command(0xA8);
        ssd1306_send_command(0x3F); // --set segment re-map 0 to 127
        ssd1306_send_command(0xD3); // --set normal display
        ssd1306_send_command(0x00); // --set multiplex ratio(1 to 64)
        ssd1306_send_command(0xD5); //
        ssd1306_send_command(0x80); // 0xa4,Output follows RAM
content;0xa5,Output ignores RAM content
        ssd1306_send_command(0xD9); // -set display offset
        ssd1306_send_command(0xF1); // -not offset
        ssd1306_send_command(0xDA); // --set display clock divide
ratio/oscillator frequency
        ssd1306_send_command(0x12); // --set divide ratio
        ssd1306_send_command(0xDB); // --set pre-charge period
        ssd1306_send_command(0x40); //
        ssd1306_send_command(0x20); // --set com pins hardware configuration
        ssd1306_send_command(0x02);
        ssd1306_send_command(0x8D); // --set vcomh
        ssd1306_send_command(0x14); // 0x20,0.77xVcc
        ssd1306_send_command(0xA4); // --set DC-DC enable
        ssd1306_send_command(0xA6); //
        ssd1306_send_command(0xAF); // --turn on oled panel
}

void ssd1306_xfer_start(void){
        DIGITAL_WRITE_HIGH(SSD1306_SCL);        // Set to HIGH
        DIGITAL_WRITE_HIGH(SSD1306_SDA);        // Set to HIGH
        DIGITAL_WRITE_LOW(SSD1306_SDA);         // Set to LOW
        DIGITAL_WRITE_LOW(SSD1306_SCL);         // Set to LOW
}
```

```
void ssd1306_xfer_stop(void){
        DIGITAL_WRITE_LOW(SSD1306_SCL);         // Set to LOW
        DIGITAL_WRITE_LOW(SSD1306_SDA);         // Set to LOW
        DIGITAL_WRITE_HIGH(SSD1306_SCL);        // Set to HIGH
        DIGITAL_WRITE_HIGH(SSD1306_SDA);        // Set to HIGH
}

void ssd1306_send_byte(uint8_t byte){
        uint8_t i;
        for(i=0; i<8; i++)
        {
                if((byte << i) & 0x80)
DIGITAL_WRITE_HIGH(SSD1306_SDA);                else
DIGITAL_WRITE_LOW(SSD1306_SDA);
DIGITAL_WRITE_HIGH(SSD1306_SCL);                DIGITAL_WRITE_LOW(SSD1306_SCL);
}       DIGITAL_WRITE_HIGH(SSD1306_SDA);        DIGITAL_WRITE_HIGH(SSD1306_SCL);
DIGITAL_WRITE_LOW(SSD1306_SCL); } void ssd1306_send_command(uint8_t command){
ssd1306_xfer_start();   ssd1306_send_byte(SSD1306_SA);  // Slave address, SA0=0
ssd1306_send_byte(0x00);        // write command
ssd1306_send_byte(command);     ssd1306_xfer_stop(); } void
ssd1306_send_data_start(void){  ssd1306_xfer_start();
ssd1306_send_byte(SSD1306_SA);  ssd1306_send_byte(0x40);        //write data }
void ssd1306_send_data_stop(void){      ssd1306_xfer_stop(); } void
ssd1306_setpos(uint8_t x, uint8_t y) {  ssd1306_xfer_start();
ssd1306_send_byte(SSD1306_SA);  //Slave address,SA0=0   ssd1306_send_byte(0x00);
//write command         ssd1306_send_byte(0xb0+y);
ssd1306_send_byte(((x&0xf0)>>4)|0x10); // |0x10
        ssd1306_send_byte((x&0x0f)|0x01); // |0x01

        ssd1306_xfer_stop();
}

void ssd1306_fillscreen(uint8_t fill_Data){
        uint8_t m,n;
        for(m=0;m<8;m++)
        {
                ssd1306_send_command(0xb0+m);   //page0-page1
                ssd1306_send_command(0x00);             //low column start
address
                ssd1306_send_command(0x10);             //high column start
address
                ssd1306_send_data_start();
                for(n=0;n<128;n++)              {
ssd1306_send_byte(fill_Data);           }
ssd1306_send_data_stop();       } } void ssd1306_char_f6x8(uint8_t x, uint8_t y,
const char ch[]){       uint8_t c,i,j=0;        while(ch[j] != '\0')    {
c = ch[j] - 32;                 if(x>126)
                {
                        x=0;
                        y++;
                }
                ssd1306_setpos(x,y);
                ssd1306_send_data_start();
```

```
                for(i=0;i<6;i++)
                {

ssd1306_send_byte(pgm_read_byte(&ssd1306xled_font6x8[c*6+i]));
                }
                ssd1306_send_data_stop();
                x += 6;
                j++;
        }
}

// Routines to set and clear bits (used in the sleep code)
#ifndef cbi
#define cbi(sfr, bit) (_SFR_BYTE(sfr) &= ~_BV(bit))
#endif
#ifndef sbi
#define sbi(sfr, bit) (_SFR_BYTE(sfr) |= _BV(bit))
#endif

void system_sleep() {
  ssd1306_fillscreen(0x00);
  ssd1306_send_command(0xAE);
  cbi(ADCSRA,ADEN);                    // switch Analog to Digitalconverter OFF
  set_sleep_mode(SLEEP_MODE_PWR_DOWN); // sleep mode is set here
  sleep_enable();
  sleep_mode();                        // System actually sleeps here
  sleep_disable();                     // System continues execution here when
watchdog timed out
  sbi(ADCSRA,ADEN);                    // switch Analog to Digitalconverter ON
  ssd1306_send_command(0xAF);
}
```

References:

- Tinusaur OLED control: https://bitbucket.org/tinusaur/ssd1306xled
- Programming the Attiny85: http://highlowtech.org/?p=1706
- Overcome sketch size issue: https://github.com/TCWORLD/ATTinyCore/tree/master/PCREL%20Patch%20for%20GCC
- Sleep the Attiny: http://www.re-innovation.co.uk/web12/index.php/en/blog-75/306-sleep-modes-on-attiny85

September 23, 2014 ilya Uncategorized

26 thoughts on “Attiny85 Breakout Keychain Game”

Georgi

January 18, 2015 at 4:41 am

Which library is included before font6x8.h ?

ilya

February 2, 2015 at 5:11 pm

Sorry, just seen this, its the eeprom library, must've been stripped out as a tag because of <>

Jacob H

February 1, 2015 at 8:56 pm

I'm having trouble compiling this. I keep getting " 'ssd1306_xxx' was not declared in this scope" for all the ssd1306_ commands. I notice that one of the #include is blank up there in your code–which library is supposed to be there? I tried #include but got the same problems. When you have a moment, can you clear up that part of your code? Thank you for this fun project!

ilya

February 2, 2015 at 5:11 pm

Sorry, just seen this, its the eeprom library, must've been stripped out as a tag because of <>

Jacob H

February 6, 2015 at 2:56 am

Thank you for your quick reply! I'm still getting the errors, but I'm trying to figure it out from the tinusaur guide there. We should #include the ssd1306xled libraries too, right? I tried adding ssd1306xled8x16.h and ssd1306xled.h but still have the same problems. Maybe I didn't put them in the right directory?

Thanks again 🙂

ilya

February 14, 2015 at 8:35 pm

I find that I can generally find a solution on google by searching for the error message you get from the compiler. Another thing to check is that you have the attiny85 board selected as it will not compile under a different board. I can email you my libraries and hardware folders if you want, email me at ilya@ilyatitov.com

---

ron wright

May 24, 2015 at 5:56 pm

I'm a newbe and I'm having trouble getting the sketch to
compile. also don't know if the board choice is correct.
I've used attiny85's on a couple of simple led dice projects and I don't know how to code unfortunately. also can you send me your library and hardware files?
Any help would be greatly appreciated.
These are the errors I'm getting:
This report would have more information with
"Show verbose output during compilation"
enabled in File > Preferences.
Arduino: 1.0.6 (Windows NT (unknown)), Board:
"ATtiny85 @ 8 MHz (internal oscillator; BOD disabled)"
attiny85_breakout_game_1.ino: In function 'void loop()':
attiny85_breakout_game_1:52: error: 'ssd1306_char_f6x8' was not declared in this scope
attiny85_breakout_game_1.ino: In function 'void resetGame()':
attiny85_breakout_game_1:196: error: 'ssd1306_char_f6x8' was not declared in this scope
attiny85_breakout_game_1.ino: In function 'void ssd1306_send_command(uint8_t)':
attiny85_breakout_game_1:355: error: 'x' was not declared in this scope

---

JACKSON

August 2, 2015 at 5:42 pm

I need your help to fix the code!

/* 2014
* Breakout game by Ilya Titov. Find building instructions on http://webboggles.com/
* The code that does not fall under the licenses of sources listed below can be used non

```
commercially with attribution.
*
* If you have problems uploading this sketch, this is probably due to sketch size – you need to update ld.exe in arduino\hardware\tools\avr\avr\bin
*
https://github.com/TCWORLD/ATTinyCore/tree/master/PCREL%20Patch%20for%20GCC
*
* This sketch is using the screen control and font functions written by Neven Boyanov for the http://tinusaur.wordpress.com/ project
* Source code and font files available at: https://bitbucket.org/tinusaur/ssd1306xled
*
* Sleep code is based on this blog post by Matthew Little:
* http://www.re-innovation.co.uk/web12/index.php/en/blog-75/306-sleep-modes-on-attiny85
*/
#include
#include

#include
#include // needed for the additional interrupt

volatile byte player = 0; //0 to 128-platformWidth – this is the position of the bounce platform
byte platformWidth = 16;
byte ballx = 62; // coordinate of the ball
byte bally = 50; // coordinate of the ball
int vdir = -1; // vertical direction and step distance
int hdir = -1; // horizontal direction and step distance
long lastFrame = 0; // time since the screen was updated last
boolean row1[16]; // on-off array of blocks
boolean row2[16];
boolean row3[16];
int score = 0; // score – counts the number of blocks hit and resets the array above when devisible by 48(total blocks)
ISR(PCINT0_vect){ // PB0 pin button interrupt
if (player >0){player–;}
return;
}
void playerInc(){ // PB2 pin button interrupt
```

```
if (player <128-platformWidth){player++;}
}

void setup() {
resetGame();
DDRB = 0b00000010; // set PB1 as output (for the speaker)
PCMSK = 0b00000001; // pin change mask: listen to portb bit 1
GIMSK |= 0b00100000; // enable PCINT interrupt
sei(); // enable all interrupts
attachInterrupt(0,playerInc,CHANGE);
lastFrame = millis();
}
void loop() {
delay(40);
noInterrupts();
SSD1306.ssd1306_init();
SSD1306.ssd1306_fillscreen(0x00);
SSD1306.ssd1306_char_f8x16(16, 4, "B R E A K O U T");
SSD1306.ssd1306_char_f8x16(20, 6, "webboggles.com");
beep(200,600); beep(300,200); beep(400,300);
delay(2000);
while (1==1) {
// continue moving after the interrupt
if (digitalRead(2)==1){if (player <128-platformWidth){player++;} if (player <128-
platformWidth){player++;} if (player 0){player–;} if (player >0){player–;} if (player >0)
{player–;}}

// bounce off the sides of the screen
if ((bally+vdir1&&vdir==-1)){bally+=vdir;}else {vdir = vdir*-1;}
if ((ballx+hdir1&&hdir==-1)){ballx+=hdir;}else {hdir = hdir*-1;}

// frame actions
if (lastFrame+1010&&bally+vdir>=54&&(ballxplayer+platformWidth)){ // game over if
the ball misses the platform
int topScore = EEPROM.read(0);
topScore = topScore <topScore){topScore = score; EEPROM.write(1,topScore & 0xFF);
EEPROM.write(0,(topScore>>8) & 0xFF); }
SSD1306.ssd1306_fillscreen(0x00);
SSD1306.ssd1306_char_f8x16(32, 3, “Game Over”);
SSD1306.ssd1306_char_f8x16(32, 5, “score:”);
```

```
char temp[4] = {0,0,0,0};
itoa(score,temp,10);
SSD1306.ssd1306_char_f8x16(70, 5, temp);
SSD1306.ssd1306_char_f8x16(32, 6, “top score:”);
itoa(topScore,temp,10);
SSD1306.ssd1306_char_f8x16(90, 6, temp);
for (int i = 0; i<1000; i++){
beep(1,random(0,i*2));
}
delay(1000);
system_sleep();
resetGame();
}else if (ballx10&&bally+vdir>=54){ // if the ball hits left of the platform bounce left
hdir=-1; beep(20,600);
}else if (ballx>player+platformWidth/2&&bally>10&&bally+vdir>=54){ // if the ball hits
right of the platform bounce right
hdir=1; beep(20,600);
}else if (bally+vdir>=54){
hdir=1; beep(20,600);
}

collisionCheck: // go back to here if a collision was detected to prevent flying through a
rigid
if (floor((bally+vdir)/8)==2){
if (row3[ballx/8]==1){row3[ballx/8]=0; score++;
collision(); goto collisionCheck; // check collision for the new direction to prevent flying
through a rigid
}
}else if (floor((bally+vdir)/8)==1){
if (row2[ballx/8]==1){row2[ballx/8]=0; score++;
collision(); goto collisionCheck;
}
}else if (floor((bally+vdir)/8)==0){
if (row1[ballx/8]==1){row1[ballx/8]=0; score++;
collision(); goto collisionCheck;
}
}

// reset blocks if all have been hit
if (score%48==0){
```

```
for (byte i =0; i<16;i++){
row1[i]=1; row2[i]=1; row3[i]=1;
}
}
}

// update whats on the screen
noInterrupts();

// blocks
SSD1306.ssd1306_setpos(0,0);
SSD1306.ssd1306_send_data_start();
for (int bl = 0; bl <16; bl++){
if(row1[bl]==1){
sendBlock(1);
}else {
sendBlock(0);
}
}
SSD1306.ssd1306_send_data_stop();
SSD1306.ssd1306_setpos(0,1);
SSD1306.ssd1306_send_data_start();
for (int bl = 0; bl <16; bl++){
if(row2[bl]==1){
sendBlock(1);
}else {
sendBlock(0);
}
}
SSD1306.ssd1306_send_data_stop();
SSD1306.ssd1306_setpos(0,2);
SSD1306.ssd1306_send_data_start();
for (int bl = 0; bl <16; bl++){
if(row3[bl]==1){
sendBlock(1);
}else {
sendBlock(0);
}
}
SSD1306.ssd1306_send_data_stop();
```

```
// clear area below the blocks
SSD1306.ssd1306_setpos(0,3);
SSD1306.ssd1306_send_data_start();
for (byte i =0; i<128; i++){
SSD1306.ssd1306_send_byte(B00000000);
}
SSD1306.ssd1306_send_data_stop();
SSD1306.ssd1306_setpos(0,4);
SSD1306.ssd1306_send_data_start();
for (byte i =0; i<128; i++){
SSD1306.ssd1306_send_byte(B00000000);
}
SSD1306.ssd1306_send_data_stop();
SSD1306.ssd1306_setpos(0,5);
SSD1306.ssd1306_send_data_start();
for (byte i =0; i<128; i++){
SSD1306.ssd1306_send_byte(B00000000);
}
SSD1306.ssd1306_send_data_stop();
SSD1306.ssd1306_setpos(0,6);
SSD1306.ssd1306_send_data_start();
for (byte i =0; i<128; i++){
SSD1306.ssd1306_send_byte(B00000000);
}
SSD1306.ssd1306_send_data_stop();
SSD1306.ssd1306_setpos(0,7);
SSD1306.ssd1306_send_data_start();
for (byte i =0; i<128; i++){
SSD1306.ssd1306_send_byte(B00000000);
}
SSD1306.ssd1306_send_data_stop();

// draw ball
SSD1306.ssd1306_setpos(ballx,bally/8);
uint8_t temp = B00000001;
SSD1306.ssd1306_send_data_start();
temp = temp << bally%8+1;
SSD1306.ssd1306_send_byte(temp);
SSD1306.ssd1306_send_data_stop();
```

```
drawPlatform();
interrupts();
//

}
}
void resetGame(){
SSD1306.ssd1306_char_f8x16(16, 4, "B R E A K O U T");
SSD1306.ssd1306_char_f8x16(20, 6, "webboggles.com");
beep(200,600); beep(300,200); beep(400,300);
delay(2000);
for (byte i =0; i<16;i++){ // reset blocks
row1[i]=1; row2[i]=1; row3[i]=1;
}
platformWidth = 16;
ballx = 64;
bally = 50;
hdir = -1;
vdir = -1;
score = 0;
player = random(0,128-platformWidth);
ballx = player+platformWidth/2;
}

void collision(){ // the collsision check is actually done befor this is called, this code
works out where the ball will bounce
if ((bally+vdir)%8==7&&(ballx+hdir)%8==7){ // bottom right corner
if (vdir==1){hdir=1;}else if(vdir==-1&&hdir==1){vdir=1;}else {hdir=1;vdir=1;}
}else if ((bally+vdir)%8==7&&(ballx+hdir)%8==0){ // bottom left corner
if (vdir==1){hdir=-1;}else if(vdir==-1&&hdir==-1){vdir=1;}else {hdir=-1;vdir=1;}
}else if ((bally+vdir)%8==0&&(ballx+hdir)%8==0){ // top left corner
if (vdir==-1){hdir=-1;}else if(vdir==1&&hdir==-1){vdir=-1;}else {hdir=-1;vdir=-1;}
}else if ((bally+vdir)%8==0&&(ballx+hdir)%8==7){ // top right corner
if (vdir==-1){hdir=1;}else if(vdir==1&&hdir==1){vdir=-1;}else {hdir=1;vdir=-1;}
}else if ((bally+vdir)%8==7){ // bottom side
vdir = 1;
}else if ((bally+vdir)%8==0){ // top side
vdir = -1;
}else if ((ballx+hdir)%8==7){ // right side
hdir = 1;
```

```
}else if ((ballx+hdir)%8==0){ // left side
hdir = -1;
}else {
hdir = hdir*-1; vdir = vdir*-1;
}

beep(30,300);

}
void drawPlatform(){
noInterrupts();
SSD1306.ssd1306_setpos(player,7);
SSD1306.ssd1306_send_data_start();
for (byte pw = 1; pw <platformWidth; pw++){SSD1306.ssd1306_send_byte(B00000011);}
SSD1306.ssd1306_send_data_stop();
interrupts();
}
void sendBlock(boolean fill){
if (fill==1){
SSD1306.ssd1306_send_byte(B00000000);
SSD1306.ssd1306_send_byte(B01111110);
SSD1306.ssd1306_send_byte(B01111110);
SSD1306.ssd1306_send_byte(B01111110);
SSD1306.ssd1306_send_byte(B01111110);
SSD1306.ssd1306_send_byte(B01111110);
SSD1306.ssd1306_send_byte(B01111110);
SSD1306.ssd1306_send_byte(B00000000);
}else {
SSD1306.ssd1306_send_byte(B00000000);
SSD1306.ssd1306_send_byte(B00000000);
SSD1306.ssd1306_send_byte(B00000000);
SSD1306.ssd1306_send_byte(B00000000);
SSD1306.ssd1306_send_byte(B00000000);
SSD1306.ssd1306_send_byte(B00000000);
SSD1306.ssd1306_send_byte(B00000000);
SSD1306.ssd1306_send_byte(B00000000);
}
}
void beep(int bCount,int bDelay){
for (int i = 0; i<=bCount; i++){digitalWrite(1,HIGH);for(int i2=0; i2<bDelay; i2++)
```

```
{__asm__("nop\n\t");}digitalWrite(1,LOW);for(int i2=0; i2<bDelay; i2++)
{__asm__("nop\n\t");}}
}
#define DIGITAL_WRITE_HIGH(PORT) PORTB |= (1 << PORT)
#define DIGITAL_WRITE_LOW(PORT) PORTB &= ~(1 << PORT)

// Routines to set and clear bits (used in the sleep code)
#ifndef cbi
#define cbi(sfr, bit) (_SFR_BYTE(sfr) &= ~_BV(bit))
#endif
#ifndef sbi
#define sbi(sfr, bit) (_SFR_BYTE(sfr) |= _BV(bit))
#endif

void system_sleep() {
SSD1306.ssd1306_fillscreen(0x00);
SSD1306.ssd1306_send_command(0xAE);
cbi(ADCSRA,ADEN); // switch Analog to Digitalconverter OFF
set_sleep_mode(SLEEP_MODE_PWR_DOWN); // sleep mode is set here
sleep_enable();
sleep_mode(); // System actually sleeps here
sleep_disable(); // System continues execution here when watchdog timed out
sbi(ADCSRA,ADEN); // switch Analog to Digitalconverter ON
SSD1306.ssd1306_send_command(0xAF);
}
```

Amal Shajan

December 12, 2015 at 2:43 am

Is it pissible to use a nokia 5110 lcd display insted of SSD1306 OLED screen.

ilya

December 17, 2015 at 11:43 pm

Hi, I’m not sure, you will have to look up the specs for the screen and adopt the code.

manuTheLeuzi

May 29, 2016 at 8:04 pm

i don't really know as well, but i guess if you want to use a 5110 display, then you should switch to a more powerful uController, i guess a AtMega328 would do perfect for that.
Hope i could help 🙂

Pingback: Attiny85 UFO Escape keychain Game | Webboggles

Francesc

January 16, 2016 at 9:24 pm

Why is the game running so slow???

ilya

February 16, 2016 at 10:58 am

Make sure you use the 8 mhz bootloader on the Attiny85

Pingback: Attiny85 Game Kit Assembly Instructions – Webboggles

Dylan Distasio

April 12, 2016 at 1:47 pm

I would be interested in getting one of your kits if you make them available. Nice job!

ilya

April 13, 2016 at 5:17 pm

Hi Dylan, I don't currently have any kits available unfortunately. I'm hoping to make a larger batch at some point, meanwhile you can make one using the files and instructions provided.

Jaldomir

April 29, 2016 at 8:19 pm

Hi Ilya, congratulations! It is a outstanding project.
I have almost same OLED display, but doesn't have I2C communication, it uses SPI (four communications pins, plus clearscreen and VCC/GND pins).
My question: there is some easy form to convert the communication sections from your code to adapt to SPI display?
Once more, congratulations!

ilya

April 29, 2016 at 9:30 pm

Thank you Jaldomir, unfortunately there are not enough pins on the attiny85 for the SPI screen + 2 buttons + speaker so there is not an east fix. You could look up reference sheet for your screen and use a microcontroller with more pins but you would have to modify the code

---

Mirko Alessandria

May 10, 2016 at 8:52 am

Excellent work ilya 🙂
I'm trying to print the pcb from your pdf, but the results aren't good. My laser printer on Xubuntu print the blak traces in gray, so use the print for toner transfer is almost impossible. Can you provide the kicad files? From Kikad I can print the pcbs in solid black, maybe the cause is the printer driver..

Thanks a lot, Mirko 🙂

ilya

May 11, 2016 at 9:25 pm

Hi Mirko, the board was created in a different tool (pcb creator), you should be able to change the print settings via the printer driver. Make sure to turn off any toner saving options.

Mirko Alessandria

May 28, 2016 at 4:42 pm

Thanks, it was the printer's driver (sorry Samsung, I'm not so ECO, lol).
Now I must how to fix the infamous library error:

"ssd1306_char_f6x8′ was not declared in this scope"

Radek

June 12, 2016 at 5:54 pm

Hi, could you please send me the HEX file with the code and programming fuses? I can not make it out of the text … Thank Radek

---

ron wright

September 3, 2016 at 10:11 pm

hello Ilya,
I,ve tried to download a couple of your sketches using the attiny85 as a pocket game. I think they're very cool.
First of all I'm not a programmer(I wish), I'm a 69 year tinkerer and love to make things. Anyway I need your help if you don't mind. here's the error report for the above sketch. I have installed all the libraries needed.
This report would have more information with
"Show verbose output during compilation"
enabled in File > Preferences.
Arduino: 1.0.6 (Windows NT (unknown)), Board: "ATtiny85 @ 8 MHz (internal oscillator; BOD disabled)"
sketch_sep03a.ino: In function 'void loop()':
sketch_sep03a:50: error: 'ssd1306_init' was not declared in this scope
sketch_sep03a:51: error: 'ssd1306_fillscreen' was not declared in this scope
sketch_sep03a:52: error: 'ssd1306_char_f6x8' was not declared in this scope
sketch_sep03a:118: error: 'ssd1306_setpos' was not declared in this scope
sketch_sep03a:119: error: 'ssd1306_send_data_start' was not declared in this scope
sketch_sep03a:127: error: 'ssd1306_send_data_stop' was not declared in this scope
sketch_sep03a:153: error: 'ssd1306_send_byte' was not declared in this scope
sketch_sep03a:159: error: 'ssd1306_send_byte' was not declared in this scope
sketch_sep03a:165: error: 'ssd1306_send_byte' was not declared in this scope
sketch_sep03a:171: error: 'ssd1306_send_byte' was not declared in this scope
sketch_sep03a:177: error: 'ssd1306_send_byte' was not declared in this scope
sketch_sep03a:186: error: 'ssd1306_send_byte' was not declared in this scope
sketch_sep03a.ino: In function 'void resetGame()':

sketch_sep03a:196: error: ‘ssd1306_char_f6x8’ was not declared in this scope
sketch_sep03a.ino: In function ‘void drawPlatform()’:
sketch_sep03a:239: error: ‘ssd1306_setpos’ was not declared in this scope
sketch_sep03a:240: error: ‘ssd1306_send_data_start’ was not declared in this scope
sketch_sep03a:241: error: ‘ssd1306_send_byte’ was not declared in this scope
sketch_sep03a:242: error: ‘ssd1306_send_data_stop’ was not declared in this scope
sketch_sep03a.ino: In function ‘void sendBlock(boolean)’:
sketch_sep03a:247: error: ‘ssd1306_send_byte’ was not declared in this scope
sketch_sep03a:256: error: ‘ssd1306_send_byte’ was not declared in this scope
sketch_sep03a.ino: In function ‘void ssd1306_send_command(uint8_t)’:
sketch_sep03a:355: error: ‘x’ was not declared in this scope

ilya

October 11, 2016 at 8:32 pm

Hi Ron, sorry about the delay, it looks like you are missing the library for controlling the screen.
There are my library folders for use with Arduino 1.0 you can find here:
https://drive.google.com/file/d/0B3bO_ZNaxxnlUGt5MnBjeEdnLU0/view?usp=sharing

---

Pingback: An ATTiny85 based handheld game | ye old Half Byte blog

---

Proudly powered by WordPress