

loulou

Looping Louie Erweiterung

Benjamin Rauser

19. Dezember 2017

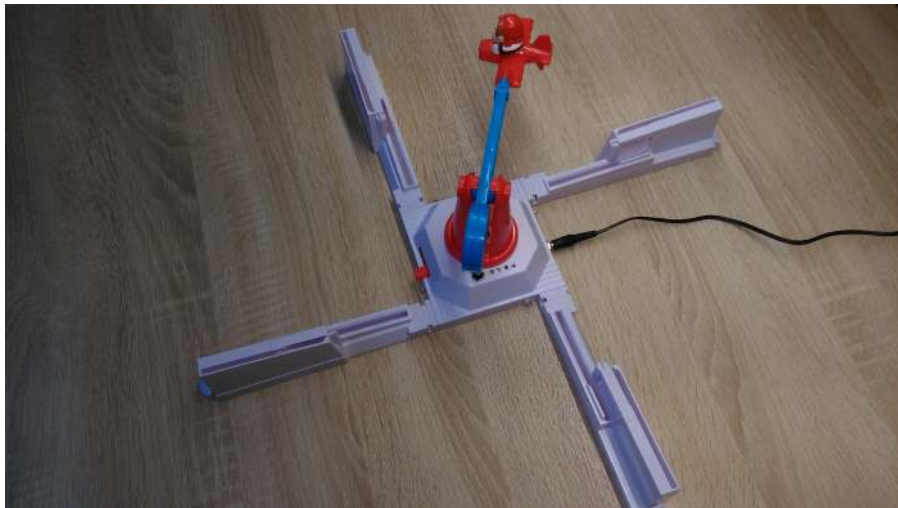


Abbildung 1: Looping Louie mit loulou Erweiterung

Inhaltsverzeichnis

1. Motivation	3
2. Werkzeug	4
3. Hinweise	4
3.1. Revision 1	4
4. Changelog	5
4.1. Revision 1	5
5. Anleitung	6
5.1. Spielbasis	6
5.2. Platine	13
5.3. Zusammenbau	17
6. Programm	19
6.1. Spielmodis	19
6.2. Source code	20
6.2.1. Github	20
6.2.2. Defines	20
6.2.3. Main-Methode	21
6.2.4. Init-Methode	22
6.2.5. Setup-Methode	23
6.2.6. Interrupt Service Routine	23
6.2.7. Calculate direction Methode	24
6.2.8. Calculate speed Methode	24
6.3. Kompilieren des Sourcecodes	26
6.3.1. Abhängigkeiten an das Buildsystem	26
6.3.2. Kompilieren	26
6.3.3. Aufräumen	26
6.3.4. Erstellen der Dokumentation	26
6.4. Wie der Attiny2313 programmiert wird	27
6.4.1. Programming	27
7. Komponenten	28
7.1. Platine	28
8. Lizenz	30
9. Haftungsausschluss	30
10. Anhang	32
A. Layout	33
B. Schaltplan	34
C. Template	35

1. Motivation

Intrinsisch

2. Werkzeug

Für den Umbau von Looping Louie werden diverse Werkzeuge benötigt. Diese sind in Table 1 aufgelistet.

Schraubenzieher (Kreuzschlitz)
Akkuschrauber
Bohrer (3/5/7/8mm)
Bügelsäge
Feile
Zange
Seitenschneider
Lötkolben

Tabelle 1: Werkzeuge

3. Hinweise

3.1. Revision 1

- Das Loch für den Drucktaster und der Befestigungsschraube sind zu klein. Diese sollte vorsichtig aufgebohrt werden.

4. Changelog

4.1. Revision 1

- Erstellen des Schaltplanes.
- Erstellen des C-Quellcodes.
- Erstellen des Platinen-Layouts.
- Erstellen der Dokumentation.
- Hinzufügen von Fotos.

5. Anleitung

5.1. Spielbasis

Zeitaufwand: 8h

Als erstes widmen wir uns der Spielbasis, zu sehen in Abbildung 2. Sie muss im ersten Schritt zerlegt und angepasst werden, um Platz für die Platine zu schaffen.

Dazu entfernen wir als erstes die 7 Schrauben an der Unterseite.

Die ersten 5 Schrauben sind mit bloßem Auge von aussen sichtbar. Die letzten zwei Schrauben sind nach dem öffnen des Batteriefaches zu sehen.

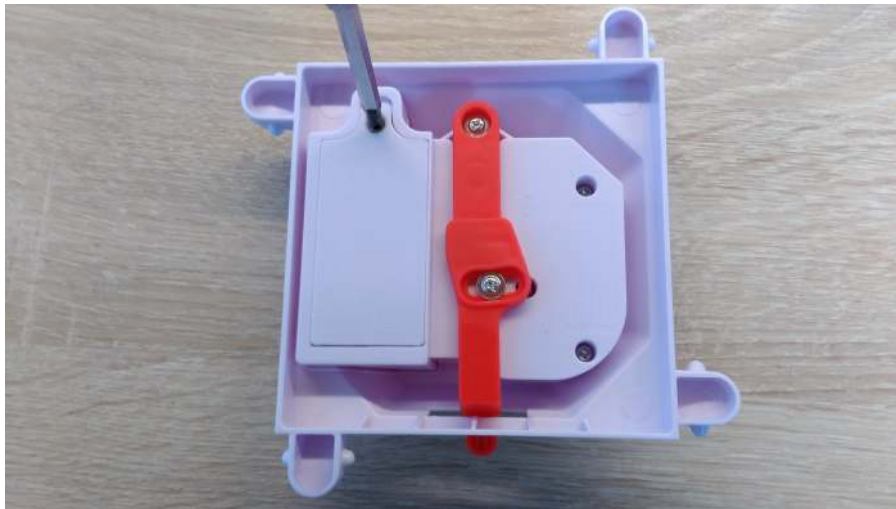


Abbildung 2: Spielbasis - Unterseite aufschrauben

Abbildung 3 zeigt die Spielbasis nach dem entfernen der 7 Schrauben.

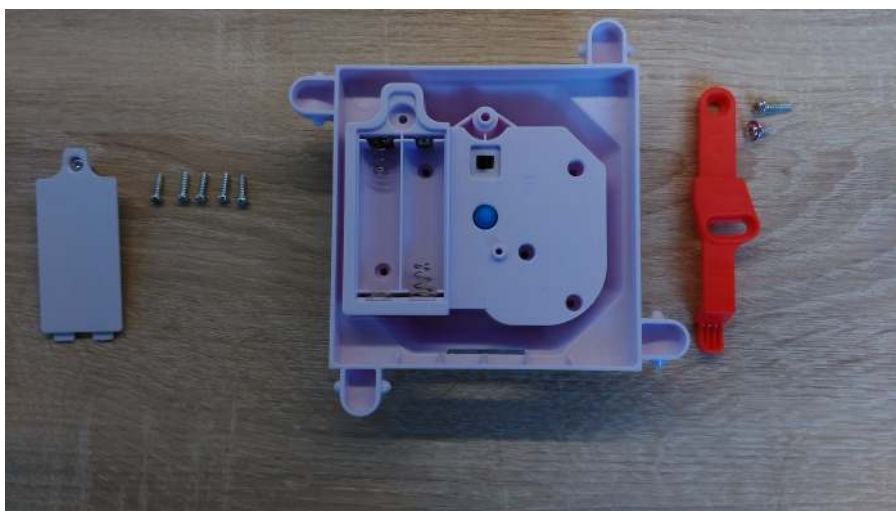


Abbildung 3: Spielbasis - geöffnet

Anschließend muss die Spielbasis geöffnet werden. Dazu zieht man das Innenteil der Spielbasis heraus. Zu sehen ist nun das Getriebe, der Motor und der Stromschalter (siehe Abbildung 4 und 5).

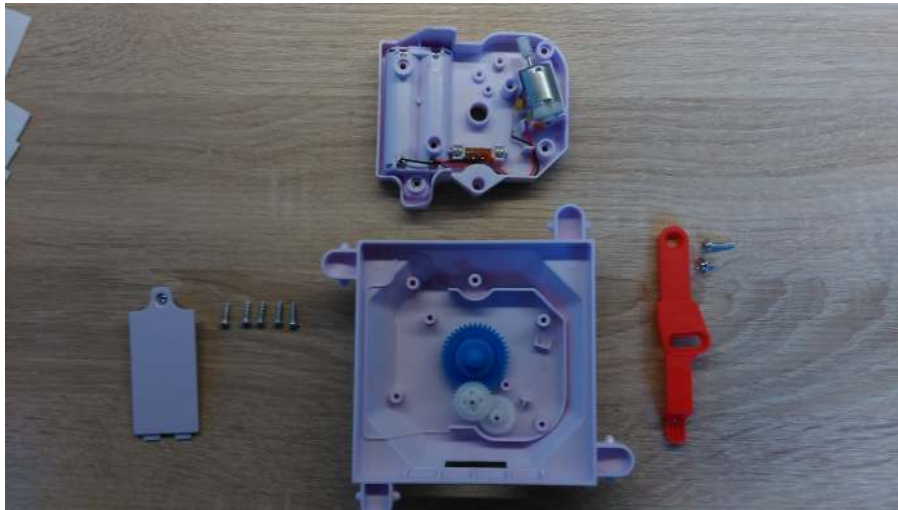


Abbildung 4: Spielbasis - geöffnet

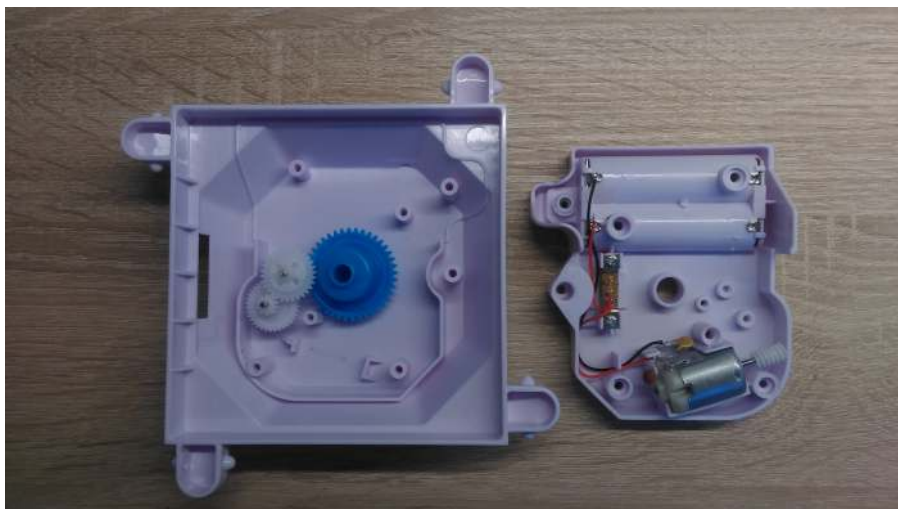


Abbildung 5: Spielbasis - Abdeckung

Nun muss der Motor, der Stromschalter und die Batteriekontakte entfernt werden. Am besten löst man hierfür zuerst die Stromkabel, mit einem Lötkolben, von den Batteriekontakten. Anschließend kann der Motor ganz einfach herausgezogen und der Stromschalter mittels eines Schraubenziehers entfernt werden. Danach können die Batteriekontakte mit einer Zange herausgezogen werden. Abbildung 6 zeigt die Abdeckung der Spielbasis ohne Motor, Stromschalter und Batteriekontakte.

Als nächstes muss Platz für die Platine geschaffen werden. Dazu bohrt man an den 4 Ecken des Batteriefaches jeweils ein Loch und sägt mit einer Säge am Rand



Abbildung 6: Spielbasis - Unterseite Abdeckung ohne Motor und Batteriekontakten

des Batteriefaches entlang (zu sehen in Abbildung 7).

Achte an dieser Stelle besonders auf das Gehäuse - Es sollte nicht brechen bzw. es sollte nicht weiter als die Umrandung des Batteriefach gesägt werden. Ansonsten sieht man beim späteren Zusammenbau der Spielbasis die Schnitte der Säge. Ebenso sollten keine Gewinde für die Schrauben abgesägt werden, mit Ausnahme der Gewinde im Batteriefach. Für die feineren Arbeiten kann eine Feile benutzt werden. Anschließend muss in den herausstehenden Seiten jeweils eine Ecke herausgefeilt werden (Gut zu sehen in der Abbildung 8). Diese werden später benötigt um die Stromkabel für die LEDs und des Netzteils aus der Spielbasis herauszuführen.



Abbildung 7: Spielbasis - Unterseite beim Aussägen

Abbildung 8 zeigt die fertig vorbereitete Unterseite der Spielbasis. Gut zu sehen sind hier die abgefeilten Ecken für die Stromkabel und das ausgesägte Batteriefach. Wichtig an dieser Stelle ist auch, dass das Batteriefach vollständig entfernt und bis auf den letzten Millimeter abgefeilt wurde. Ansonsten kann es vorkommen, dass das Gehäuse beim späteren Zusammenbau nicht richtig schließt.

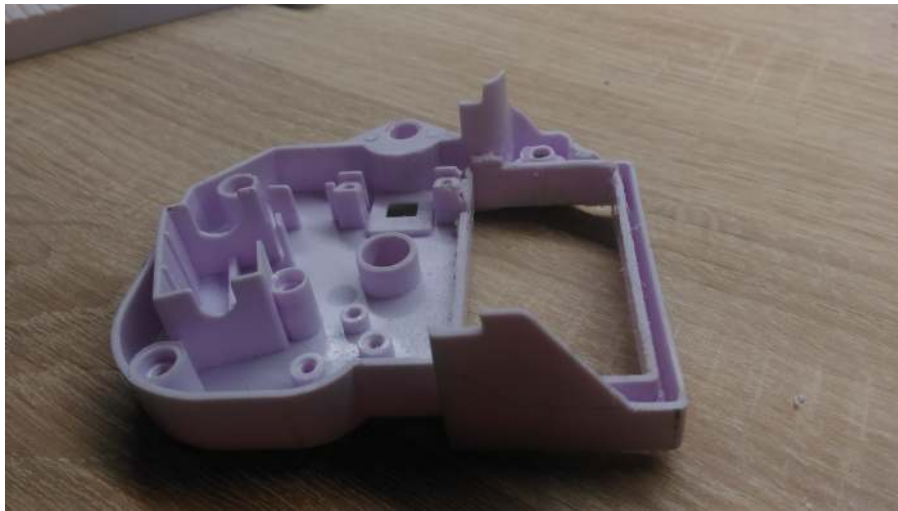


Abbildung 8: Spielbasis - Unterseite ausgesägt und geschliffen

Nun kann die Unterseite der Spielbasis zur Seite gelegt werden. Es folgt die Vorbereitung der Oberseite.

Als erstes sollte hier das Gewinde im linken oberen Eck (siehe Abbildung 9) mit einem Bohrer entfernt werden. Dieses Gewinde ist der Platine im Weg und muss daher entfernt werden. Achte hierbei darauf das nicht durch das Oberteil gebohrt wird, lasse lieber ein bis zwei Millimeter überstehen.



Abbildung 9: Spielbasis - Oberseite mit Gewinde das entfernt werden muss

Als nächstes schneidet man zuerst die Schablonen (siehe Anhang 10) aus. Diese werden benötigt um die Löcher für die LEDs, den Taster und den Stromanschluß richtig zu positionieren. Abbildung 10 zeigt die Schablonen und das fertig Vorbereitete Gehäuse.

Im nächsten Schritt nimmt man den große ausgeschnitten Teil der Schablone und legt ihn auf die

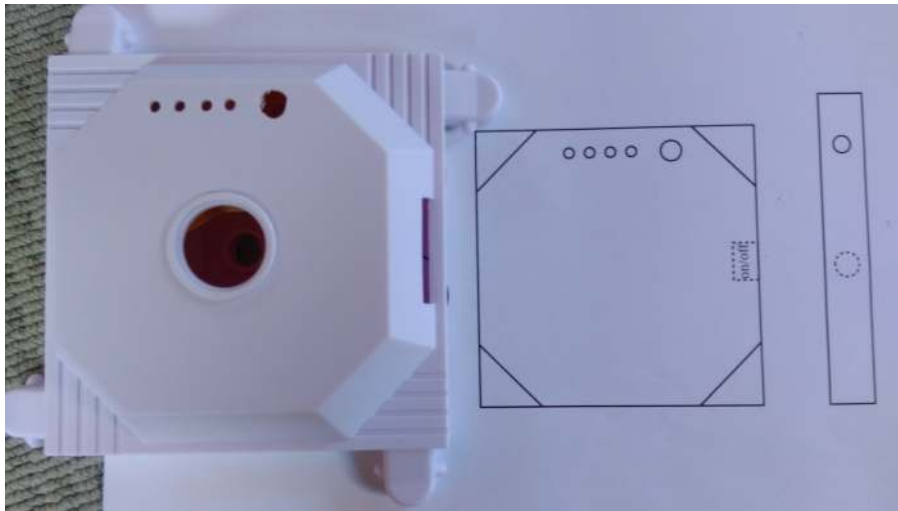


Abbildung 10: Schablonen und Oberteil der Spielbasis

Oberseite der Spielbasis. Man sollte hier darauf achten dass die Schrift “on/off” über dem AN/AUS-Schalter der Spielbasis liegt (zu sehen in Abbildung 11). Anschließend können zuerst mit einem spitzen Gegenstand (zum Beispiel einem Bohrer) die Löcher auf dem Oberteil der Spielbasis angeköhrt werden. Die kleinen Löcher, für die LEDs, werden nun mit einem 3mm-Bohrer durchbohrt. Im Anschluss wird das Loch für den Drucktaster mit einem 7mm-Bohrer in das Oberteil gebohrt. Abbildung 11 zeigt die gebohrten Löcher in dem Oberteil der Spielbasis. Zu sehen ist hier auch dass die Löcher auf der rechten Seite, vom AN/AUS-Schalter aus gesehen, liegen.



Abbildung 11: Spielbasis - Gebohrte Löcher für die LEDs und den Taster im Oberteil

Nun nimmt man das kleine rechteckige Teil der Schablone und körnt damit an jeder der 4 Seiten das Loch für die 5mm großen LEDs an (Kreis mit durchgezogener Linie). Dabei muss der Kreis mit der durchgezogenen Linie zu dem Verbindungsstück, für die Arme des Looping Louie, zeigen (zu sehen in Abbildung 12). Der gestrichelte Kreis ist für den Stromanschluss. Dieser wird auf der, dem des AN/AUS-Schalter gegenüberliegenden Seite, gebohrt (zu sehen in Abbildung 12). Die Löcher für die LEDs werden mit einem Bohrer der Stärke von 5mm und das Loch für den Stromanschluss mit einen 7mm großen Bohrer gebohrt.



Abbildung 12: Spielbasis - Mit gebohrten Löchern in der Seite des Oberteils

Zum Abschluss können nun noch alle LEDs, der Drucktaster und der Stromanschluss in die Spielbasis gesteckt werden. Sollte hier noch etwas klemmen kann an dieser Stelle noch leicht nachgearbeitet werden.

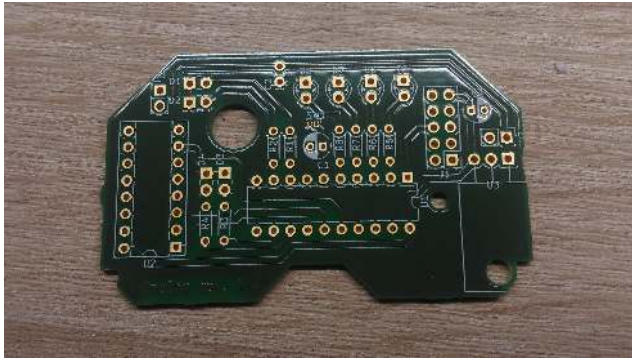


Abbildung 13: Spielbasis - Mit gesteckten LEDs, dem Drucktaster und dem Stromanschluss

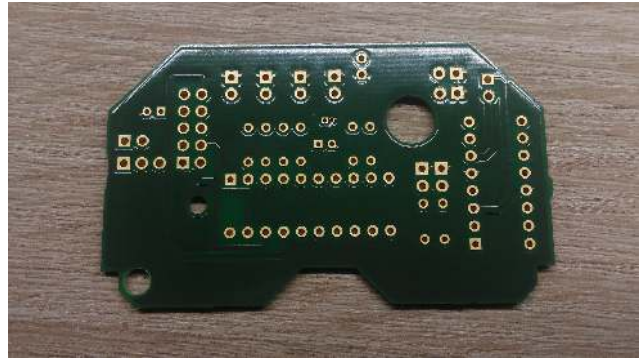
Die Abbildung 13 zeigt die Spielbasis mit den gesteckten LEDs, dem Drucktaster und dem Stromanschluss. Die Bauteile können nun wieder aus der Spielbasis entfernt und zur Seite gelegt werden.

5.2. Platine

Nun widmen wir uns der Platine - dem Herzstück der Erweiterung von Looping Louie. Zu sehen ist diese in den Abbildungen 14 (a) und (b).



(a) Oberseite



(b) Unterseite

Abbildung 14: Originale Platine - Revision 1

Auf der Platine sind unter anderem ein kleiner Microcontroller (ATTINY2313), ein Treiberbaustein, ein Festspannungsregler sowie diverse LEDs und Widerstände verbaut. Der Schaltplan 10 und das Platinen-Layout 10 kann im Anhang gefunden werden. Die Referenzen der Bauteile können der Tabelle 4 entnommen werden.

Am einfachsten beginnt man an dieser Stelle mit den IC-Sockeln (U1 und U2), sowie dem Festspannungsregler (U3). Im Anschluss kann der 10polige Wannenstecker (J1), für den Programmer, aufgelötet werden, dieser ist optional und wird nur benötigt wenn man den Attiny zum programmieren nicht herausnehmen möchte. Nachdem die IC-Sockel und der Festspannungsregler aufgelötet wurden, können die Widerstände (R1-R8) und die Kondensatoren (C1-C3) aufgelötet werden. Die Abbildung 15 zeigt die Platinen nach dem auflöten der Widerstände, der Kondensatoren, dem Festspannungsregler und den IC-Sockeln.

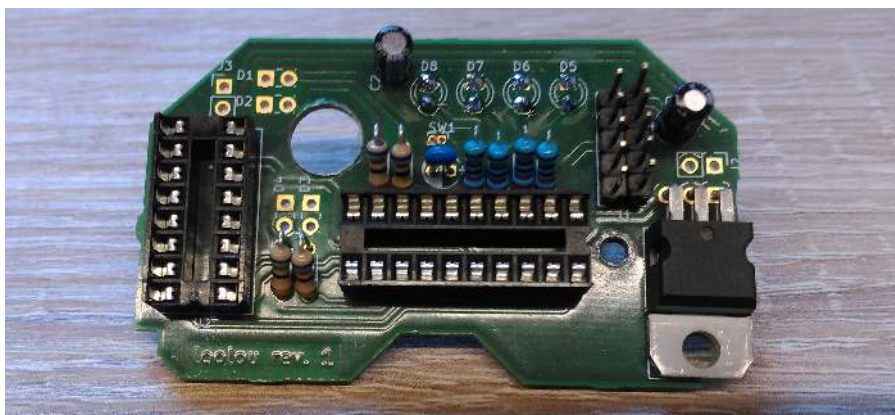


Abbildung 15: Platine mit aufgelöteten Widerständen und IC-Sockel

Nachdem nun die IC-Sockel (U1,U2), der Festspannungsregler (U3), die Widerstände (R1-R8) und die Kondensatoren (C1-C3) aufgelötet wurden, sollten die LEDs (D5-D8) aufgelötet werden. Dazu steckt man diese einzeln, von innen, in das Oberteil der Spielbasis und schraubt dann die Platine in den dafür vorgesehenen Platz im Oberteil der Spielbasis. Wie die Platine in das Oberteil der Spielbasis geschraubt wird zeigt die Abbildung 16

Nachdem nun die LEDs in die Spielbasis gesteckt und die Drähte der LEDs (D5-D8) durch die dafür vorgesehenen Löcher geführt wurden, können diese an der Platine festgelötet werden. Achte an dieser Stelle darauf das die LEDs richtig gepolt sind, ansonsten müssen diese später wieder entfernt und neu aufgelötet werden.



Abbildung 16: Platine festgeschraubt im Oberteil der Spielbasis

Nachdem nun auch die LEDs D5 bis D8 aufgelötet sind kann die Platine nun wieder aus der Spielbasis herausgenommen werden. Die Abbildung 17 zeigt die Platine mit den festgelöteten LEDs (D5-D8). Es empfiehlt sich an dieser Stelle die bisher festgelöteten Bauteile und die Lötstellen mit einem Durchgangsprüfer zu testen, um kalte Lötstellen und verpolte LEDs rechtzeitig zu erkennen.

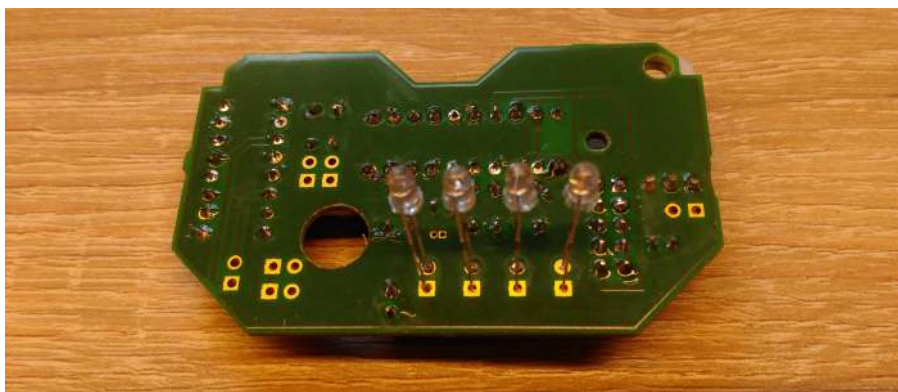


Abbildung 17: Platine mit LEDs auf der Unterseite

An dieser Stelle sollten die Litzen für die restlichen LEDs (D1-D4), den Drucktaster (SW1), den Hohlstecker-Buchse (J2) und den Motor vorbereitet werden. Die Länge der Litzen kann einfach der Tabelle 2 entnommen werden. Es werden jeweils eine rote und eine schwarze Litze benötigt.

Reference	Description	Length
D1	LED1, 5mm	9cm
D2	LED2, 5mm	13cm
D3	LED3, 5mm	13cm
D4	LED4, 5mm	13cm
SW1	Drucktaster	5cm
J1	Hohlstecker-Buchse	7cm
J3	Motor	16cm

Tabelle 2: Litzenlänge

Sobald diese vorbereitet wurden, können die Litzen bereits an die LEDs (D1-D4) gelötet werden. Dazu schneidet man am Besten die Drähte der LEDs auf einen Zentimeter herunter und lötet dann die Kabel an. Im Anschluss sollten noch die offenen Lötstellen mit einem Schrumpfschlauch verschlossen werden. Anschließend können auch die Litzen an den Drucktaster und die Hochstecker-Buchse gelötet und mit einem Schrumpfschlauch verschlossen werden.

Das Pin-Layout, der Hohlstecker-Buchse, kann dem Datenblatt entnommen werden. Für die Hohlstecker-Buchse "HEBL 21" (www.reichelt.com) ist das Pin-Layout in der Abbildung 18 zu sehen.

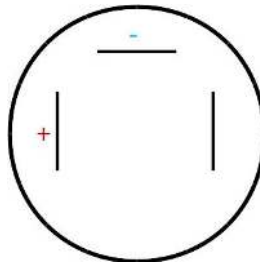


Abbildung 18: Pin-Layout Hohlstecker-Buchse "HEBL 21"

Die alten Litzen des Motors werden nun durch die neuen längeren Litzen ersetzt. Dabei muss die rote Litze einmal durchtrennt und wie gehabt mit dem Stromschalter verbunden werden.

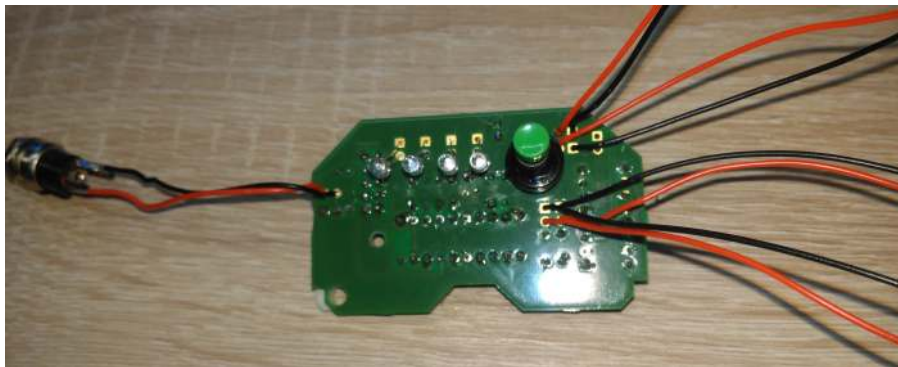
Im Anschluss wird der Drucktaster mit den angelöteten Litzen, von unten in das große Loch der Platine (zwischen R2 und D2) gesteckt. Die Litzen führt man dazu ebenfalls von unten nach oben durch das Loch. Die beiden Litzen (rot und schwarz) werden nun an den beiden Löchern von SW1 von oben festgelötet. Die Polung spielt hier keine Rolle.

Sobald der Drucktaster SW1 an der Platine festgelötet wurde, können die auch die LEDs (D1-D4), die Hohlstecker-Buchse (J2) und der Motor (J3) festgelötet werden. Hierbei muss auf die Polung der LEDs, der Hohlstecker-Buchse und des Motors geachtet werden - die schwarzen Litzen (Minus-Pol) müssen in die Löcher mit den quadratischen Kontakten. Im Fall einer Verpolung kann es zu Schäden

an der Platine, zu nicht funktionierenden LEDs oder einen Motor der in die falsche Richtung dreht kommen. In der Abbildung 19 ist die fertig gelötete Platine zu erkennen (ohne Motor).



(a) Oberseite



(b) Unterseite

Abbildung 19: Platine mit allen Bauteilen (ohne Motor)

Nun sollte die Platine nochmals mit dem Durchgangsprüfer, auf kalte Lötstellen und verpolte Bauteile, überprüft werden. Ist hier kein Fehler zu finden, dann kann zum Testen das Netzteil angeschlossen werden. Wird der Festspannungsregler schnell sehr heiß dann deutet dies auf einen Kurzschluss oder eine Verpolung hin. Weiterhin sollte zwischen den Pins 1 und 2 des Festspannungsregler eine Spannung von 9 Volt und zwischen Pin 2 und Pin 3 eine Spannung von 5V anliegen.

Im Anschluss kann der Attiny2313 und der Treiberbaustein (L293D) auf die Sockel gesteckt werden. Die halbrunden Einkerbungen müssen dafür über den weißen halbrunden Kreisen des Zeichnung (bei U1 und U2) liegen.

Steckt man nun wieder das Netzteil ein, dann sollten die LEDs D1 bis D5 abwechselnd und die LED D8 dauerhaft leuchten. Der Motor kann über den Stromschalter, zwischen J3 und Motor, an bzw. ausgeschaltet werden. Durch den Druck auf den Drucktaster (SW1) können die LEDs D5 bis D7 getestet werden.

Ist bisher kein Fehler erkennbar, dann kann man die Platine ausschalten und zur Seite legen.

5.3. Zusammenbau

Nun werden alle Komponenten wieder zusammengebaut. Beginnen muss man beim Zusammenbau mit dem Getriebe. Dazu nimmt man das blaue Zahnrad und steckt es in das große Loch des Oberteils der Spielbasis. Anschließend müssen die weißen Zahnräder eingesetzt werden. Nachdem diese eingesetzt wurden, wird die Platine in das Oberteil geschraubt, dazu führt man die Litzen der LEDs, des Motors und der Hohlstecker-Buchse an der linken bzw. rechten Seite heraus. Der Stromschalter des Motors und der Motor selbst werden wieder in die Unterseite der Spielbasis eingesetzt bzw. festgeschraubt. Wie dies am Ende aussehen muss zeigt die Abbildung 20.

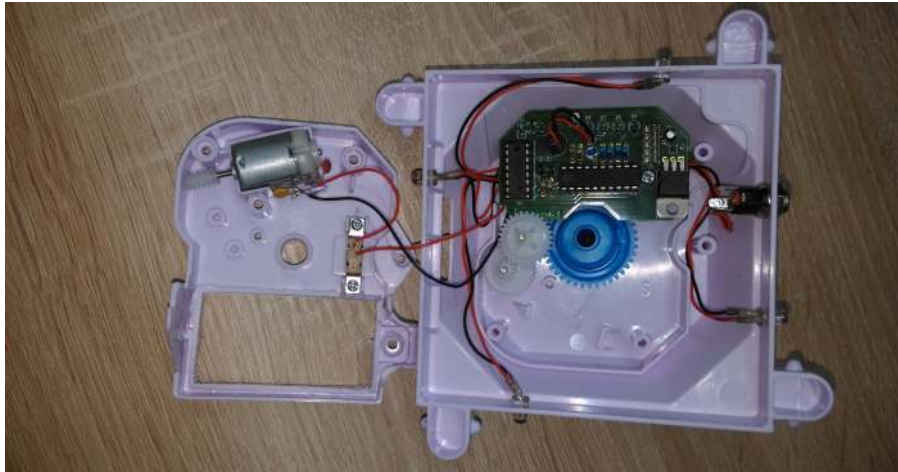


Abbildung 20: Spielbasis mit eingesetzter Platine

Im Anschluss wird die Unterseite auf die Oberseite gesetzt und die 5 Schrauben der Spielbasis wieder festgeschraubt. Nun sollte man prüfen, ob die Spielbasis wieder richtig schließt. Sollte beim Vorbereiten der Spielbasis nicht sauber gearbeitet worden sein, dann muss hier nochmals nachgearbeitet werden. Die Spielbasis darf keine Erhebungen aufweisen, ansonsten steht die Spielbasis nicht sicher oder der Arm des Looping Louie kann nicht richtig arretiert werden. Die Abbildung 21 zeigt die wieder zusammengebaute Spielbasis ohne Batteriefachabdeckung.

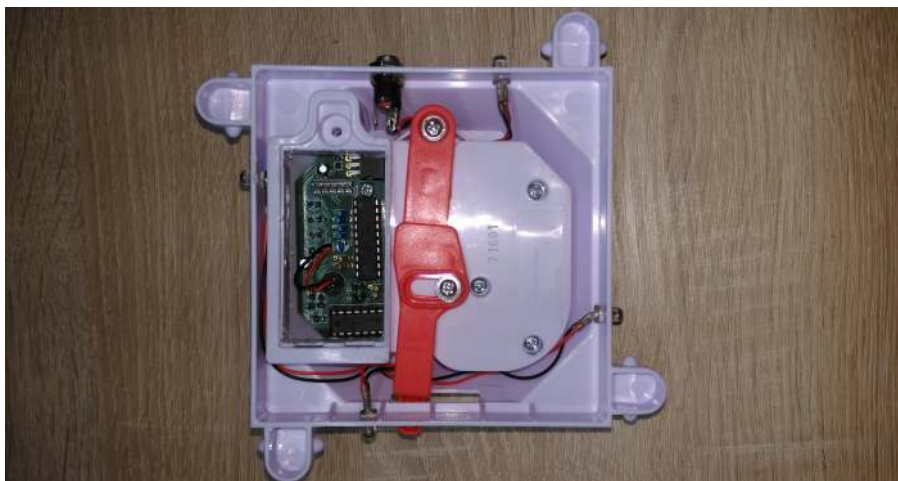
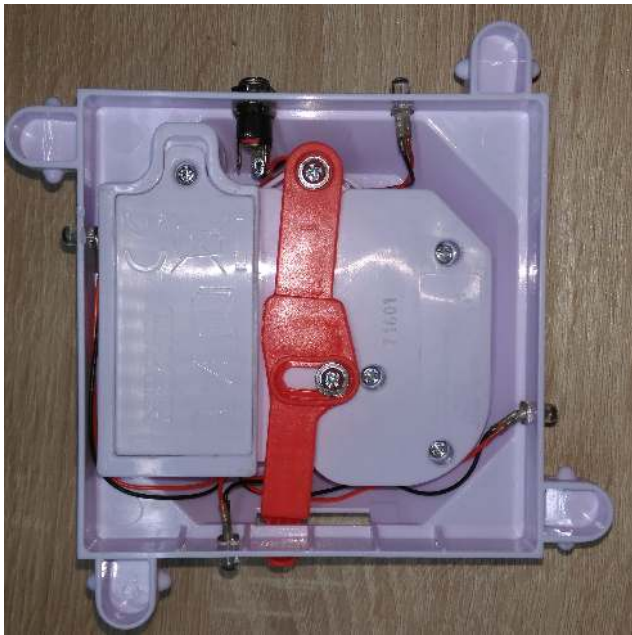


Abbildung 21: Zusammengebaute Spielbasis ohne Batteriefachabdeckung

In der folgenden Abbildung (Abbildung 22) ist die geschlossene Spielbasis von oben und unten zu sehen.



(a) Oberseite



(b) Unterseite

Abbildung 22: Platine mit allen Bauteilen

6. Programm

6.1. Spielmodis

Die loolou Erweiterung hat vier verschiedene Modi zum Spielen. Diese können während dem Spiel einfach über den Drucktaster am Oberteil der Spielbasis umgeschaltet werden. Die vier roten LEDs neben dem Drucktaster zeigen dabei den aktuellen Modus an.

Die folgende Tabelle beschreibt die 4 verschiedenen Modi.


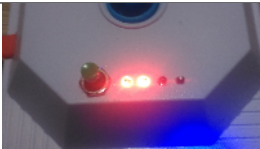


Modus	Beschreibung
	Einfacher Modus. Looping Louie dreht sich so schnell wie beim originalen Spiel.
	Schneller Modus. Looping Louie dreht sich viel schneller als gewohnt.
	Zufälliger Modus. Looping Louie ändert seine Geschwindigkeit und bleibt auch mal stehen.
	Zufälliger Modus mit Rückwärtsgang. Looping Louie ändert seine Geschwindigkeit und Richtung.

Tabelle 3: Spielmodi

6.2. Source code

Der Source-Code von loolou ist veröffentlicht und kann heruntergeladen und selbst kompiliert werden. Bitte beachten Sie die Lizenz von loolou. Zu finden ist diese im Kapitel 8 auf Seite 30.

6.2.1. Github

Der Source-Code ist auf Github veröffentlicht und kann unter folgendem Link heruntergeladen werden.

<https://github.com/berauser/loolou>

6.2.2. Defines

Die defines sind in der Datei main.c definiert. Sie definieren die PINs und PORTs der LEDs, PWMs und des Drucktasters für den ATTINY2313. Ebenso werden hier die Timer-Delays und PWM-Frequenzen für den Spielablauf definiert. Das folgende Listing (Listing 1) zeigt alle, für den Spielablauf wichtigen, Defines.

Listing 1: Defines

```
// *****
// PWM values
// *****
#define PWM_OFF    0 // 0V ( OFF )
#define PWM_3V     85 // 3V ( (9V / 255) * 85 = 3V )
#define PWM_9V    255 // 9V

// *****
// PWM values
// *****
//
// 1000000Hz ( F_CPU )
// ----- = 976,56 ( 0x3D0 )
// 1024 * 1Hz (1 sec)
#define TIMER_INT_DEPLAY 0x3D0

// *****
// Pin description
// *****
//
// attiny 2313
//
// -----
// RESET    -| 1 (PA2)   (VCC) 20 |- VCC
// LED5      -| 2 (PD0)   (PB7) 19 |- SCK
// LED6      -| 3 (PD1)   (PB6) 18 |- MISO
// LED7      -| 4 (PA1)   (PB5) 17 |- MOSI
// LED8      -| 5 (PA0)   (PB4) 16 |- FORWARD
// BUTTON    -| 6 (PD2)   (PB3) 15 |- BACKWARD
// NC        -| 7 (PD3)   (PB2) 14 |- PWM
// LED1      -| 8 (PD4)   (PB1) 13 |- LED4
// LED2      -| 9 (PD5)   (PB0) 12 |- LED3
// GND       -| 10(GND)  (PD6) 11 |- NC
//
// -----
//
```

```

// *** LEDs *****
#define PIN_LED1 PD4
#define PIN_LED2 PD5
#define PIN_LED3 PB0
#define PIN_LED4 PB1
#define PIN_LED5 PD0
#define PIN_LED6 PD1
#define PIN_LED7 PA1
#define PIN_LED8 PA0

#define PORT_LED1 PORTD
#define PORT_LED2 PORTD
#define PORT_LED3 PORTB
#define PORT_LED4 PORTB
#define PORT_LED5 PORTD
#define PORT_LED6 PORTD
#define PORT_LED7 PORTA
#define PORT_LED8 PORTA

#define DDR_LED1 DDRD
#define DDR_LED2 DDRD
#define DDR_LED3 DDRB
#define DDR_LED4 DDRB
#define DDR_LED5 DDRD
#define DDR_LED6 DDRD
#define DDR_LED7 DDRA
#define DDR_LED8 DDRA

// *** DIRECTION / PWM *****
#define PIN_DIRECTION_FORWARD PB4
#define PIN_DIRECTION_BACKWARD PB3
#define PIN_PWM PB2

#define PORT_DIRECTION_FORWARD PORTB
#define PORT_DIRECTION_BACKWARD PORTB
// #define PORT_PWM

#define DDR_DIRECTION_FORWARD DDRB
#define DDR_DIRECTION_BACKWARD DDRB
#define DDR_PWM DDRB

#define REGISTER_PWM OCR0A
#define MODE_PWM (1 << COM0A1) | (1 << WGM00)
#define CLOCK_PWM (1 << CS01)

#define REGISTER_TIMER OCR1A
#define MODE_TIMER_A 0x00
#define MODE_TIMER_B (1 << WGM12) | (1 << CS12) | (1 << CS10) // clk/1024

// *** BUTTON *****
#define PIN_BUTTON PD2
#define PORT_BUTTON PORTD
#define DDR_BUTTON DDRD
#define EDGE_TYPE_BUTTON (1 << ISC01) // interrupt on falling edge

```

6.2.3. Main-Methode

Im Listing 2 ist die main-Funktion des Programmes. Diese initialisiert die Peripherie und springt dann in eine while-true-loop. Die while-true-loop setzt in jeden Durchlauf die aktuelle Geschwindigkeit und

schaltet die LEDs für die Richtungsanzeige ein, beziehungsweise aus. Anschließend werden 250ms gewartet bis die nächste LED für die Richtungsanzeige eingeschaltet wird.

Listing 2: Main-Method

```
// *****  
// game  
// *****/  
int main(void) {  
  
    init();  
    setup();  
  
    /* main loop */  
    while (1) {  
        /* all other uses interrupts */  
        trigger_speed( speed );  
        trigger_direction( direction );  
        show_direction( direction );  
        _delay_ms( 250 );  
    }  
  
    return 0;  
}
```

6.2.4. Init-Methode

Die Methode init initialisiert die GPIOs (General Porpuse Input/Output) des ATTINY.

Listing 3: Setup-Method

```
// *****  
// initialize gpio, set pin directions  
// *****/  
void init( void )  
{  
  
    // leds  
    GPIO_init( DDR_LED5, PIN_LED5, OUTPUT ); // 1  
    GPIO_init( DDR_LED6, PIN_LED6, OUTPUT ); // 2  
    GPIO_init( DDR_LED7, PIN_LED7, OUTPUT ); // 3  
    GPIO_init( DDR_LED8, PIN_LED8, OUTPUT ); // 4  
  
    GPIO_init( DDR_LED1, PIN_LED1, OUTPUT ); // red  
    GPIO_init( DDR_LED2, PIN_LED2, OUTPUT ); // blue  
    GPIO_init( DDR_LED3, PIN_LED3, OUTPUT ); // green  
    GPIO_init( DDR_LED4, PIN_LED4, OUTPUT ); // yellow  
  
    // motor control  
    GPIO_init( DDR_DIRECTION_BACKWARD, PIN_DIRECTION_BACKWARD, OUTPUT ); //  
        backward  
    GPIO_init( DDR_DIRECTION_FORWARD, PIN_DIRECTION_FORWARD, OUTPUT ); //  
        forward  
    GPIO_init( DDR_PWM, PIN_PWM, OUTPUT ); // PWM  
    PWM_enable( MODE_PWM, CLOCK_PWM, REGISTER_PWM, PWM_OFF );  
  
    // button
```

```

GPIO_init( DDR_BUTTON, PIN_BUTTON, INPUT);
GPIO_interrupt( PORT_BUTTON, PIN_BUTTON, INTO, EDGE_TYPE_BUTTON );

// timer interrupt
TIMER_enable( MODE_TIMER_A, MODE_TIMER_B );

// Enable interrupts
sei();
}

```

6.2.5. Setup-Methode

Die Setup-Methode initialisiert das loolou-Programm mit den default-Werten.

Listing 4: Init-Method

```

// *****
// setup syStem with default values
// *****/
void init( void )
{
    // set initial values
    active_led = LED1;
    mode       = M_NORMAL_FORWARD;
    speed      = PWM_3V;
    direction  = FORWARD;

    // show current mode
    show_mode();

    // set seed
    init_random( TCNT1L );

    // initial pwm
    trigger_speed( PWM_3V );

    // set initial timer delay
    TIMER_set( TIMER_INT_DISPLAY );
}

```

6.2.6. Interrupt Service Routine

Das Programm loolou implementiert zwei verschiedene Interrupt Service Routinen (ISR) - die ISR(INT0_vect) und ISR(TIMER1_COMPA_vect).

Die ISR(INT0_vect) wird bei einer fallenden Flanke am Drucktaster aufgerufen. In der ISR(INT0_vect) wird der Modus des Spieles um eins hochgezählt und neu ausgegeben. Anschließend wird der Timer zurückgesetzt und aufgerufen um die neue Geschwindigkeit und Spiel-Richtung zu setzen.

Die Interrupt Service Routine ISR(TIMER1_COMPA_vect) wird jede Sekunde aufgerufen und berechnet die Geschwindigkeit und die Spiel-Richtung neu.

Listing 5: Interrupt Service Routine

```

// *****

```

```

// interrupt service routine
// *****/
ISR(INT0_vect)
{
    ADD_ONE_BETWEEN( mode, M_NORMAL_FORWARD, M_RANDOM_RANDOM );
    show_mode();

    /* reset timer, call ISR to change speed/direction immediately */
    TIMER_reset();
    TIMER1_COMPA_vect();
}

ISR(TIMER1_COMPA_vect)
{
    direction = calc_direction();
    speed = calc_speed();
}

```

6.2.7. Calculate direction Methode

Die Funktion `calc_direction` gibt anhand des aktuellen Spielmodus die Spiel-Richtung zurück. Für die drei Spielmodis “Einfacher Modus“, “Schneller Modus“ und “Zufälliger Modus“ wird immer die Spiel-Richtung “forwärts“ ausgegeben. Im Spielmodus “Zufälliger Modus mit Rückwärtsgang“ wird anhand einer Zufallszahl die Spiel-Richtung berechnet. In diesem Modus wird etwa 10% “rückwärts“ und etwa 90% “forwärts“ zurückgegeben.

Listing 6: Calculate direction

```

// *****/
// calculates a new direction
// *****/
GameDirection calc_direction( void )
{
    switch( mode )
    {
        case M_NORMAL_FORWARD:
        case M_FAST_FORWARD:
        case M_RANDOM_FORWARD:
        default:
            return FORWARD;
            break;
        case M_RANDOM_RANDOM:
            return ( (get_random_between( 0, 10 ) == 0) ? BACKWARD :
                    FORWARD ); /* 10% backward : 90% forward */
            break;
    }
}

```

6.2.8. Calculate speed Methode

Die Funktion `calculate_speed` gibt anhand des Spielmodus die berechnete Geschwindigkeit zurück. Im Modus “Einfacher Modus“ wird immer der PWM-Wert für 3V und im Modus “Schneller Modus“ der PWM-Wert für 9V zurückgeliefert. In den Spielmodis “Zufälliger Modus“ und “Zufälliger Modus mit Rückwärtsgang“ wird die Geschwindigkeit anhand zweier Zufallszahlen berechnet. Dabei wird die

erste Zufallszahl für die grobe Einordnung der Geschwindigkeit verwendet und die zweite Zufallszahl für die exakten PWM-Wert, der dann auch zurückgegeben wird. Die erste Zufallszahl ist in dieser Berechnung notwendig um größere Sprünge zwischen den Geschwindigkeiten zu reduzieren. Die Werte sind wie folgt aufgeteilt: Etwa 10% zwischen 0.0V und ~3.5V, etwa 70% zwischen ~3.5V und ~7.0V, sowie 20% zwischen ~7.0V und ~9.0V.

Listing 7: Calculate Speed

```
// *****
// calculates a new speed
// *****/
uint8_t calc_speed( void )
{
    switch( mode )
    {
        case M_NORMAL_FORWARD:
        default:
            return PWM_3V;
            break;
        case M_FAST_FORWARD:
            return PWM_9V;
            break;
        case M_RANDOM_FORWARD:
        case M_RANDOM_RANDOM:
        {
            /*
             * Calculate speed in 25 steps.
             * 0 = 0V, ..., 25 = ~9V
             *
             * For a better and faster playing pleasure,
             * a probability of
             * 10% values between 0V and 3,5V and
             * 70% values between 3,5V and 6,0V is selected.
             * 20% values between 7,0V and 9,0V is selected.
             */
            static uint8_t r = 0;
            r = get_random_between( 0, 10 );
            if ( r == 0 )
            { /* 20% slow ( 0V - ~3,5V ) */
                return get_random_between( 0, 100 );
            }
            else if ( r == 1 || r == 2 )
            { /* 20% very fast /~7,0V - 9,0V */
                return get_random_between( 201, 255 );
            }
            else
            { /* 70% fast ( ~3,5V - ~7,0V ) */
                return get_random_between( 101, 200 );
            }
        }
        break;
    }
}
```

6.3. Kompilieren des Sourcecodes

6.3.1. Abhängigkeiten an das Buildsystem

Um den C-Sourcecode für den ATTINY2313 zu übersetzen werden einige wenige Anforderungen an das Buildsystem gestellt. Es wird lediglich die AVR-Toolchain (avr-gcc) und ein Programmierer für Atmel-Mikrocontroller benötigt (avrdude). Um die Dokumentation zu erstellen wird zusätzlich ein LaTeX-Kompilier benötigt, unter Ubuntu oder Debian bietet sich hier zum Beispiel texlive an.

Für Ubuntu und Debian können die folgenden Befehle, zum Installieren der Abhängigkeiten, verwendet werden.

Listing 8: Build dependencies

```
# install avr toolchain and avr programmer
sudo apt-get install gcc-avr avrdude
# install LaTeX-Compiler
sudo apt-get install texlive-full
```

6.3.2. Kompilieren

Das Kompilieren des C-Sourcecodes durch einen einfachen Make-Aufruf angestoßen werden. Dazu muss im Root-Verzeichnis des Sourcecodes folgender Befehl eingegeben werden.

Listing 9: Compile

```
# compile loolou
make
```

6.3.3. Aufräumen

Um die Kompilierten Source wieder zu löschen kann der folgende Befehl verwendet werden.

Listing 10: Clean

```
# clean loolou sources
make clean
```

6.3.4. Erstellen der Dokumentation

Das Erstellen der Dokumentation kann ebenso wie das Kompilieren des Sourcecodes durch einen Make-Aufruf ausgeführt werden. Dazu fügt man das Schlüsselwort “doc“ dem “make“ an.

Listing 11: Create documentation

```
# create documentation
make doc
```

```
# clean documentation
make doc-clean
```

6.4. Wie der Attiny2313 programmiert wird

6.4.1. Programming

Ebenso wie das Kompilieren der Source kann der Attiny mittels eines vordefinierten Make-Aufrufs programmiert werden. Dies kann über den Befehl “make burn“ ausgeführt werden. Mittels den Variablen **AVRDUDE_PROGRAMMER** und **AVRDUDE_PORT** kann das Device und der Programmertyp definiert werden. Die Default-Einstellungen sind **avrispv2** für den Typ des Programmers und **/dev/ttyAMC0** für das Device. Der Typ des Programmers kann der Dokumentation des Programmers entnommen werden.

Achtung: Sollte der Programmierer eine eigene Stromversorgung für den Attiny haben, dann darf der 9V Stromanschluss des Looping Louie nicht angeschlossen werden, wenn das Programmieren über die eingebaute Programmier-Schnittstelle (J1) stattfindet. Andernfalls kann der Programmierer beschädigt werden.

Listing 12: Programming

```
# flash the attiny (using default values)
make burn
# or
make AVRDUDE_PROGRAMMER=avrispv2 AVRDUDE_PORT=/dev/ttyAMC0 burn
# AVRDUDE_PROGRAMMER define programmer type
# AVRDUDE_PORT define programmer device
```

7. Komponenten

In der folgenden Tabelle (Tabelle 4) sind alle benötigten Komponenten für die Looping Louie Erweiterung aufgelistet. Diese können über <https://www.reichelt.com> bezogen werden. Die Platine muss separat über <https://aisler.net/> bestellt werden. Der Bestelllink für die Platine steht im Kapitel 7.1 auf Seite 28.

Reference	Description	-	-	
U1	IC-Sockel, 20polig	GS 20	0,25 €	
U2	IC-Sockel, 16polig	GS 16	0,18 €	
U3	Festspannungsregler	LF50CV	0,18 €	
J1	Wannenstecker, 10polig, gerade	WSL 10G	0,09 €	
J2	Hohlstecker-Buchse Ø2.1mm	HEBL 21	0,27 €	
J3	Motor-Anschluss		-	
SW1	Drucktaster		1,04 €	
C1	Keramik-Kondensator, 0.1µF		0,26 €	
C2	Elektrolyt-Kondensator, 0.1µF		0,06 €	
C3	Elektrolyt-Kondensator, 2.2µF		0,17 €	
D1	LED, 5mm, rot		0,36 €	
D2	LED, 5mm, gruen		0,53 €	
D3	LED, 5mm, blau		0,53 €	
D4	LED, 5mm, gelb		0,41 €	
D5	LED, 3mm, rot		0,21 €	
D6	LED, 3mm, rot		0,21 €	
D7	LED, 3mm, rot		0,21 €	
D8	LED, 3mm, rot		0,21 €	
R1	Widerstand, 68Ω		0,09 €	
R2	Widerstand, 68Ω		0,09 €	
R3	Widerstand, 68Ω		0,09 €	
R4	Widerstand, 68Ω		0,09 €	
R5	Widerstand, 120Ω		0,09 €	
R6	Widerstand, 120Ω		0,09 €	
R7	Widerstand, 120Ω		0,09 €	
R8	Widerstand, 120Ω		0,09 €	
-	Microcontroller ATTINY2313-20PU	ATTINY 2313V20PU	3,45 €	
-	Kupferlitze isoliert, 90cm, 1x0,14mm, rot			
-	Kupferlitze isoliert, 90cm, 1x0,14mm, schwarz			
-	Steckernetzteil, 9V	HNP 06-090L6	7,20 €	
-	loulou-Platine			

Tabelle 4: Components

7.1. Platine

Das Platinen-Layout wurde mittels des Open-Source Tools kicad erstellt. Beziehen kann man die Platine einfach und dazu noch günstig über Aisler - eine niederländische Firma die unter anderem ihre Platinen in Deutschland herstellen lässt. Allerdings lassen sich hier die Platinen nur als ein vielfaches

von drei bestellen. Was aber in den meisten Fällen trotzdem noch günstiger ist als vergleichbare Unternehmen.

Die Platine ist bereits bei Aisler hinterlegt und kann über den folgenden Link bestellt werden.

<https://aisler.net/berauser/loulou/main-pcb>

8. Lizenz

loulou ist lizenziert unter der Charityware. Sie können es verwenden und kopieren, so viel Sie wollen, aber ich möchte Sie ermutigt, für ein Kinder-Wohltätigkeitsprojekt Ihrer Wahl zu spenden.

loulou is Charityware. You can use and copy it as much as you like, but you are encouraged to make a donation to a childrens's charity project of your choice.

9. Haftungsausschluss

Ich übernehme keinerlei Haftung für jeglichen Schaden (Schaden an Gegenständen, Personen oder jeglichen anderen Schaden), welcher in Zusammenhang mit der der Benutzung und den Umbau durch die von mir bereitgestellte Looping Louie-Erweiterung "loulou" entsteht.

Ich übernehme keinerlei Gewähr für die Aktualität, Korrektheit, Vollständigkeit oder Qualität der bereitgestellten Informationen. Haftungsansprüche gegen mich, die sich auf Schäden materieller oder ideeller Art beziehen, welche durch die Nutzung oder Nichtnutzung der dargebotenen Informationen, Programmen etc. bzw. durch die Nutzung fehlerhafter und unvollständiger Informationen verursacht wurden sind grundsätzlich ausgeschlossen, sofern kein nachweislich vorsätzliches oder grob fahrlässiges Verschulden durch mich vorliegt.

Abbildungsverzeichnis

1.	Looping Louie mit loolou Erweiterung	1
2.	Spielbasis - Unterseite aufschrauben	6
3.	Spielbasis - geöffnet	6
4.	Spielbasis - geöffnet	7
5.	Spielbasis - Abdeckung	7
6.	Spielbasis - Unterseite Abdeckung ohne Motor und Batteriekontakten	8
7.	Spielbasis - Unterseite beim Aussägen	8
8.	Spielbasis - Unterseite ausgesägt und geschliffen	9
9.	Spielbasis - Oberseite mit Gewinde das entfernt werden muss	9
10.	Schablonen und Oberteil der Spielbasis	10
11.	Spielbasis - Gebohrte Löcher für die LEDs und den Taster im Oberteil	10
12.	Spielbasis - Mit gebohrten Löchern in der Seite des Oberteils	11
13.	Spielbasis - Mit gesteckten LEDs, dem Drucktaster und dem Stromanschluss	11
14.	Originale Platine - Revision 1	13
15.	Platine mit aufgelöteten Widerständen und IC-Sockel	13
16.	Platine festgeschraubt im Oberteil der Spielbasis	14
17.	Platine mit LEDs auf der Unterseite	14
18.	Pin-Layout Hohlstecker-Buchse "HEBL 21"	15
19.	Platine mit allen Bauteilen (ohne Motor)	16
20.	Spielbasis mit eingesetzter Platine	17
21.	Zusammengebaute Spielbasis ohne Batteriefachabdeckung	17
22.	Platine mit allen Bauteilen	18

Tabellenverzeichnis

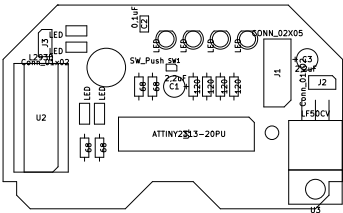
1.	Werkzeuge	4
2.	Litzenlänge	15
3.	Spielmodi	19
4.	Components	28

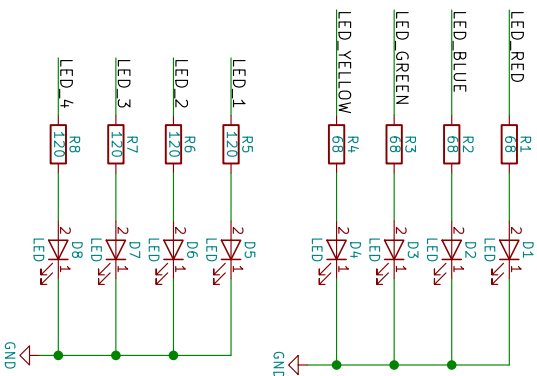
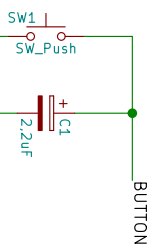
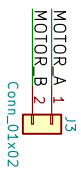
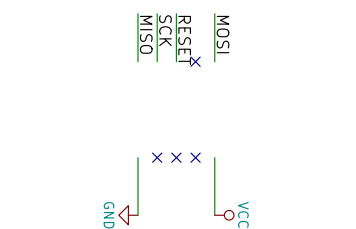
Listings

1.	Defines	20
2.	Main-Method	22
3.	Setup-Method	22
4.	Init-Method	23
5.	Interrupt Service Routine	23
6.	Calculate direction	24
7.	Calculate Speed	25
8.	Build dependencies	26
9.	Compile	26
10.	Clean	26
11.	Create documentation	26
12.	Programming	27

10. Anhang

A. Layout





C. Template

