
light_snake

Release 2.0

Guenther Beulen

Apr 22, 2021

CONTENTS:

1	Introduction	1
2	Hardware	3
3	Software	5
4	Communication	7
4.1	Bluetooth	7
4.2	Serial Communication	7
5	Class LightSnake	9
6	Class Led	11
7	Class SpeedControl	15
8	Indices and tables	17
	Index	19

INTRODUCTION

This light sculpture was build in 2013/2014. It is the one of the first LED projects from [Klingdesign](#).

Klingdesign is the light artist Christiane Kling. She produces tailor made light sculptures. Since 2013 the electronic engineer Günther Beulen has been joining Klingdesing. He develops Hard- and Software. And this is their first big project:

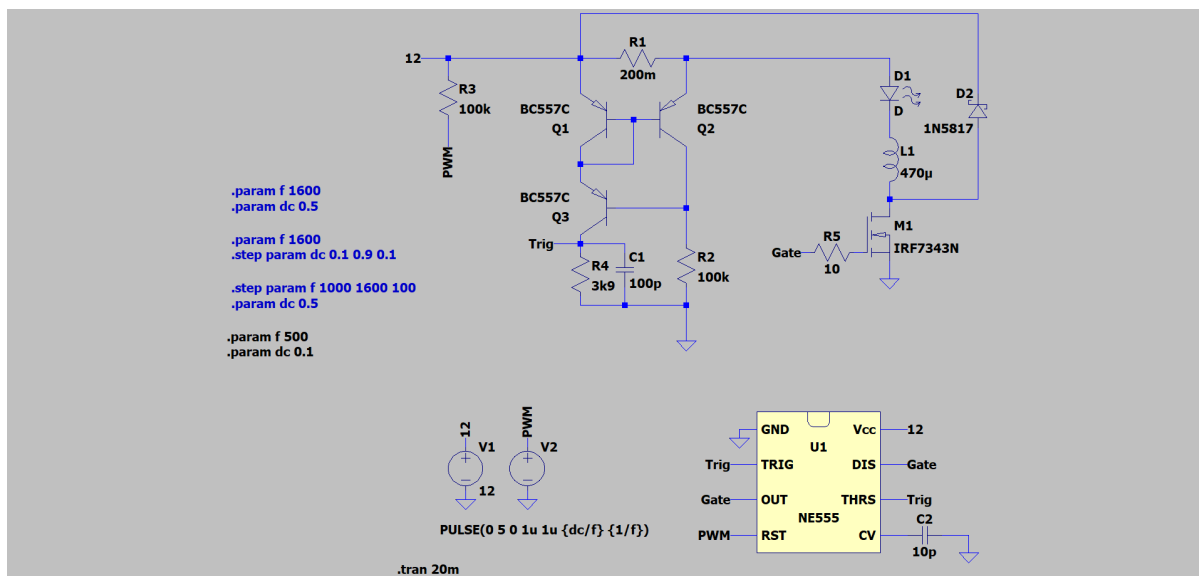


To see the light snake in action, here is the [Youtube video](#).

HARDWARE

This light sculpture includes 15 power LEDs. Each power LED has its own [current source](#).

Each current source contains a NE555 working as a two-level controller. The average value of the current is 300 mA. The pin 4 of a NE555 is a reset pin, which can be used for intensity control via PWM. The next photo shows the LTspice model, which is available in the repository.



To control all 14 current sources the [Adafruit PCA9685 16-Channel Servo Driver](#), based on the IC [PCA9685](#), is used. This IC is controlled by an [Arduino Uno](#). This board is based on the [ATmega328P](#).

SOFTWARE

The software was written with the [Arduino IDE](#). The library [Adafruit-PWM-Servo-Driver-Library](#) handles the [PCA9685](#). It is available from the Arduino library manager.

COMMUNICATION

4.1 Bluetooth

4.2 Serial Communication

The serial communication is possible with the USB port of the Arduino Uno or with the RX/TX pins of the controller.

It is possible to debug the light sculpture. Each LED can be tested. If it is currently on the PWM control may be defective and if it is currently off than the LED or the current source may be defective.

CLASS LIGHTSNAKE

class LightSnake

Public Functions

void **setup** ()

This function initializes the *LightSnake* class.

The class contains one instance of the class *Adafruit_PWMServoDriver* and an array of the class *Led*. The random generator and the class *Serial* will be initialized, too.

void **loop** ()

This function repeats the updates of the intensities of the leds.

Every element of the *Led* Class is called. Their intensities are updated and at the min or max value a new duration of the speed is evaluated. The new intensities are send via the I2C bus to the PCA9685.

void **help** ()

This method prints an info.

The Serial Monitor of the Arduino IDE, PuTTY or picocom can be used.

void **info** ()

This methode prints the info of all Leds.

The number, intensity, darker and duration of all LEDs are printed. And the time each the last loop has needed and the cycle time of the loop.

void **clearAllLEDs** ()

Clear all LEDs.

All LEDs are cleared by sending the intensity Zero to each channel of the PCA9685

int8_t **getNumber** ()

This method is used by getting the number of a LED and by getting the new duration.

Returns number

void **getLEDNumber** ()

Gets the number of the LED to test.

Get the number of the LED to test. Hexadecimal numbers are used (0..9, A, B, C, D). X will delete this loop.

void **testLED** (uint8_t)

This method tests the specified LED.

First all LED are turned off. After waiting for a second, the specified LED is turned on for a second and then turned off. So the hardware can be tested. The Serial Monitor of the Arduino IDE, PuTTY or picocom can be used.

void **testAllLEDs** ()

This method tests the LEDs and their current sources.

Every LED is being tested after this method is called. First all LED are turned off. After waiting for a second, every LED is turned on for a second and then turned off. After waiting for a second, the next LED is turned on and off. So the hardware can be tested.

void **invertOutputOfLoopDuration** ()

This method handles the output of the loop time.

The output of the looptime, the duration of an cycle, can be enabled or disabled. Every time when this method is called, the corresponding boolean variable is inverted.

void **changeLoopDuration** (bool)

This method sets a new duration time for the loop.

The time sets the duration for the loop. If this value is too small, the function will not wait and start immediately with the next cycle. After each cycle the method waits till the duration of a cycle is over. A loop duration can be set with a char. This character represents a hexadecimal digit. Hexadecimal numbers are used (0..9, A, B, C, D, E, F). The new duration time is this digit multiplied by 5 ms. At the end of the loop, the output of millis() is polled. If the value 0 is chosen, the next loop starts immediately.

void **setIndex** (bool)

This method sets the beginning of an array.

There can be different arrays containing the intensities. With this method the index of the program is set globally for all LEDs.

void **readEeprom** ()

This method reads the content of the EEPROM.

The user can set the cycle time and the index of the used PROGRAM. The setup function can read the stored values. If a 255 is read, then the EEPROM has not been programmed yet. Then the cycle time and the index are set to default values.

void **writeEeprom** ()

This method writes the current values to the EEPROM.

If the user changed the loop time and the PROGRAM index, he can save this to the EEPROM. So they will be loaded at the next start.

CLASS LED

class Led

Public Functions

uint8_t getNumber ()

Returns the number of the LED

void setNumber (uint8_t)

Parameters number – of the LED

uint16_t getIntensity ()

Returns intensity of the LED

void setIntensity (uint16_t)

Parameters intensity – of the LED

uint8_t getPointer ()

Returns pointer to the intensity table

void setPointer (uint8_t)

Parameters pointer – to the intensities

uint8_t getProgmemIndex ()

Returns pointer to the intensity table

void setProgmemIndex (uint8_t)

Parameters pointer – to the intensities

bool getDarker ()

Returns darker of the LED

void setDarker (bool)

Parameters darker – if the brightness of the LED decreases

void invertDarker ()

inverts darker

bool getWaitAtMinIntensity ()

Returns waitAtMinIntensity

void setWaitAtMinIntensity (bool)

Parameters **waitAtMinIntensity** – this LED waits at its minimal brightness

bool **getWaitAtMaxIntensity** ()

Returns waitAtMaxIntensity

void **setWaitAtMaxIntensity** (bool)

Parameters **waitAtMaxIntensity** – this LED waits at its maximal brightness

uint8_t **getCyclesAtMinIntensity** ()

Returns cyclesAtMinIntensity of the LED

void **setCyclesAtMinIntensity** (uint8_t)

Parameters **cyclesAtMinIntensity** – this LED waits this cycles at its minimal brightness

uint8_t **getCyclesAtMaxIntensity** ()

Returns cyclesAtMaxIntensity this LED waits at its minimal brightness

void **setCyclesAtMaxIntensity** (uint8_t)

Parameters **cyclesAtMaxIntensity** – this LED waits this cycles at its maximal brightness

bool **getDarkerHasChanged** ()

Returns _darkerHasChanged

bool **getIntensityAtMin** ()

Returns _intensityAtMin

bool **getIntensityAtMax** ()

Returns _intensityAtMax

void **increaseIntensity** ()

Increases the intensity.

If the new intensity is equal to the maximal intensity, the intensities will be decreased in the next step.

void **decreaseIntensity** ()

Decreases the intensity.

If the new intensity is equal to the minimal intensity, the intensities will be increased in the next step.

void **changeIntensity** ()

Increases or decreases the member _intensity.

In dependency of the boolean value of darker the method increaseIntensity or decreaseIntensity is called.

void **increasePointer** ()

Increases the pointer to the intensities.

If the new value of the pointer is equal to the size of the array, the pointer will be decreased in the next step;

void **decreasePointer** ()

Decreases the pointer to the intensities.

If the new value of the pointer is equal to zero, the pointer will be increased in the next step.

void **changePointer** ()

Increases or decreases the pointer.

In dependency of the boolean value of darker the method increasePointer or decreasePointer is called.

void **pointer2int** ()

The array with the intensities is included in the header file 'intensities.h'. This file has been created by the python script 'progmen_creator.py'.

Returns the content of the PROGMEM array.

bool **letSpeedControlCount** ()

Call counter method of class *SpeedControl*.

The property counter of the aggregated class *SpeedControl* is decreased. If its value is equal to zero, true is returned. The variable is initialized with the duration.

void **setSpeedControlDuration** (uint8_t)

Parameters **duration** – time at one intensity (property of the class *SpeedControl*)

uint8_t **getSpeedControlDuration** ()

Returns duration (the time at one intensity, property of the class *SpeedControl*)

void **setSpeedControlCounter** (uint8_t)

Parameters **counter** – If a longer duration is wanted, the counter can be set to a value greater than duration.

uint8_t **getSpeedControlCounter** ()

Returns counter counts from duration to zero

CLASS SPEEDCONTROL

class SpeedControl

Public Functions

uint8_t **getCounter** ()

Returns counter

void **setCounter** (uint8_t)

Parameters **counter** – of the LED

uint8_t **getNumber** ()

Returns number of the LED

void **setNumber** (uint8_t)

Parameters **number** – of the LED

uint8_t **getDuration** ()

Returns duration of the intensity

void **setDuration** (uint8_t)

The duration specifies the time at an intensity

Parameters **duration** – of the intensity

bool **count** ()

The method decreases the counter, if the counter has the value zero, true is returned.

Returns (counter == 0)

INDICES AND TABLES

- `genindex`
- `modindex`
- `search`

L

Led (*C++ class*), 11
 Led::changeIntensity (*C++ function*), 12
 Led::changePointer (*C++ function*), 12
 Led::decreaseIntensity (*C++ function*), 12
 Led::decreasePointer (*C++ function*), 12
 Led::getCyclesAtMaxIntensity (*C++ function*), 12
 Led::getCyclesAtMinIntensity (*C++ function*), 12
 Led::getDarker (*C++ function*), 11
 Led::getDarkerHasChanged (*C++ function*), 12
 Led::getIntensity (*C++ function*), 11
 Led::getIntensityAtMax (*C++ function*), 12
 Led::getIntensityAtMin (*C++ function*), 12
 Led::getNumber (*C++ function*), 11
 Led::getPointer (*C++ function*), 11
 Led::getProgmemIndex (*C++ function*), 11
 Led::getSpeedControlCounter (*C++ function*), 13
 Led::getSpeedControlDuration (*C++ function*), 13
 Led::getWaitAtMaxIntensity (*C++ function*), 12
 Led::getWaitAtMinIntensity (*C++ function*), 11
 Led::increaseIntensity (*C++ function*), 12
 Led::increasePointer (*C++ function*), 12
 Led::invertDarker (*C++ function*), 11
 Led::letSpeedControlCount (*C++ function*), 13
 Led::pointer2int (*C++ function*), 12
 Led::setCyclesAtMaxIntensity (*C++ function*), 12
 Led::setCyclesAtMinIntensity (*C++ function*), 12
 Led::setDarker (*C++ function*), 11
 Led::setIntensity (*C++ function*), 11
 Led::setNumber (*C++ function*), 11
 Led::setPointer (*C++ function*), 11
 Led::setProgmemIndex (*C++ function*), 11
 Led::setSpeedControlCounter (*C++ function*), 13

Led::setSpeedControlDuration (*C++ function*), 13
 Led::setWaitAtMaxIntensity (*C++ function*), 12
 Led::setWaitAtMinIntensity (*C++ function*), 11
 LightSnake (*C++ class*), 9
 LightSnake::changeLoopDuration (*C++ function*), 10
 LightSnake::clearAllLEDs (*C++ function*), 9
 LightSnake::getLEDNumber (*C++ function*), 9
 LightSnake::getNumber (*C++ function*), 9
 LightSnake::help (*C++ function*), 9
 LightSnake::info (*C++ function*), 9
 LightSnake::invertOutputOfLoopDuration (*C++ function*), 10
 LightSnake::loop (*C++ function*), 9
 LightSnake::readEeprom (*C++ function*), 10
 LightSnake::setIndex (*C++ function*), 10
 LightSnake::setup (*C++ function*), 9
 LightSnake::testAllLEDs (*C++ function*), 10
 LightSnake::testLED (*C++ function*), 9
 LightSnake::writeEeprom (*C++ function*), 10

S

SpeedControl (*C++ class*), 15
 SpeedControl::count (*C++ function*), 15
 SpeedControl::getCounter (*C++ function*), 15
 SpeedControl::getDuration (*C++ function*), 15
 SpeedControl::getNumber (*C++ function*), 15
 SpeedControl::setCounter (*C++ function*), 15
 SpeedControl::setDuration (*C++ function*), 15
 SpeedControl::setNumber (*C++ function*), 15