Smart SightLight

By Moritz von der Brake 31698, Hendrik Sensken 33359 and Berkay Kiris 34862

Introduction

by Moritz von der Brake

Many people use a night light to help them navigate their home in the dark. However there are many issues with the common night light. One issue is that a regular nightlight is static and dose not move. This means that the night light either does not have a large field of illumination or is very broad and lights up everything. The problem is made even worse by the fact that conventional nightlights attach directly to a wall socket. This is where our project the Smart Sightlight comes in. Our nightlight actively waits for a person to enter the vicinity. Once a person is in the vicinity the nightlight scans the room and calculates position and direction of movement of the person. It then uses this information to illuminate the light in-front of the person.

Method

1.CAD

by Moritz von der Brake

In order to house all the electronics in a space friendly and secure way it is necessary to design a housing that can be easily printed and assembled. This was done using OnShape which is free for all students. The housing is made of three main pieces. The center piece houses the two PIR sensors at a 45 degree angle from the wall in both directions. This means the PIR sensors have a very large field of view allowing for early detection. The center piece also has standoffs with screw holes which allows the PIR sensors to be securely attached to the 3d print. The other two main pieces are the two housings for the servos. These housing were designed with an extruding attachment part. this part allowed the servos to slide into place and then be attached using the screw holes which were also included in the 3D design. Lastly there where also to caps printed for the top and the bottom. Theses where made to allow the servos to have enough space for the TOF sensor and the light to be attached to. Furthermore it was made sure that there would be enough space for the 3D print not to interfere with the movement of either of the servos.

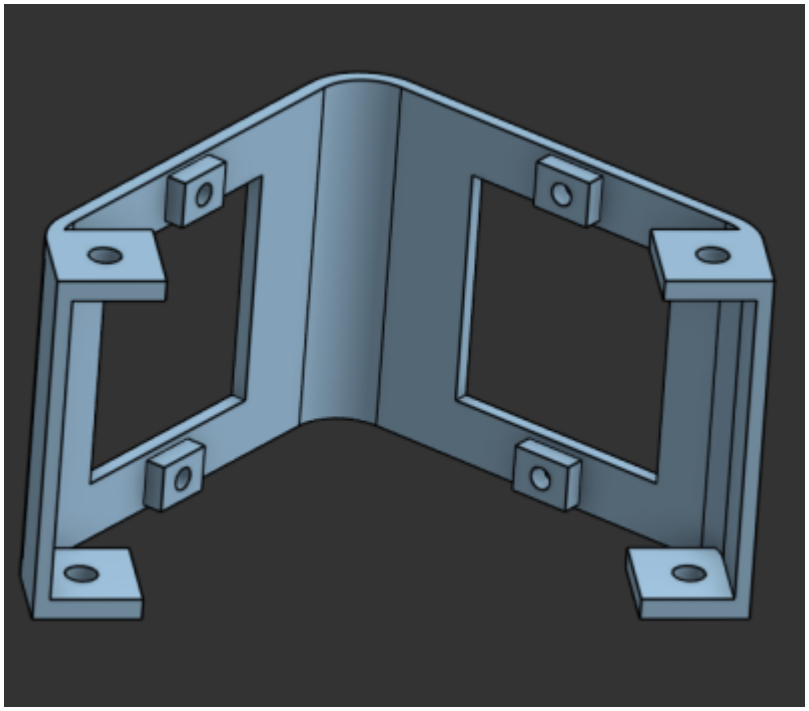


Figure 1: Picture of the PIR housing

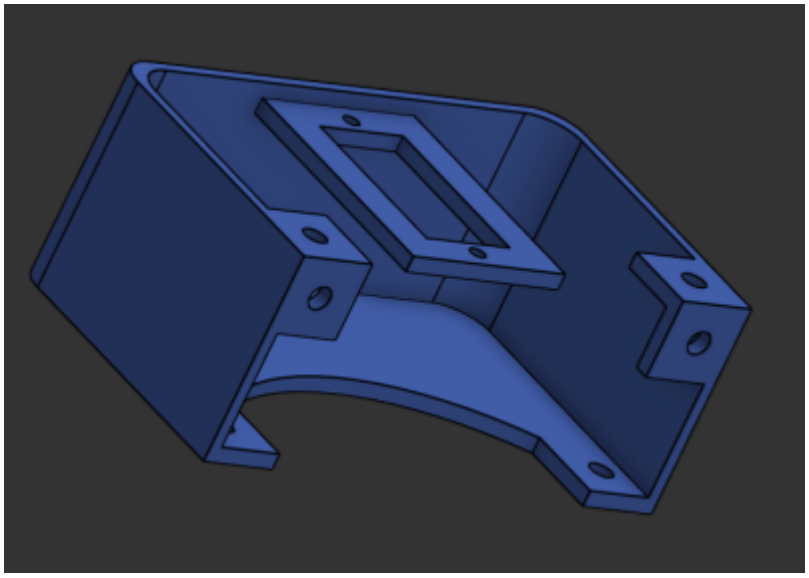


Figure 2: Picture of the servo housing

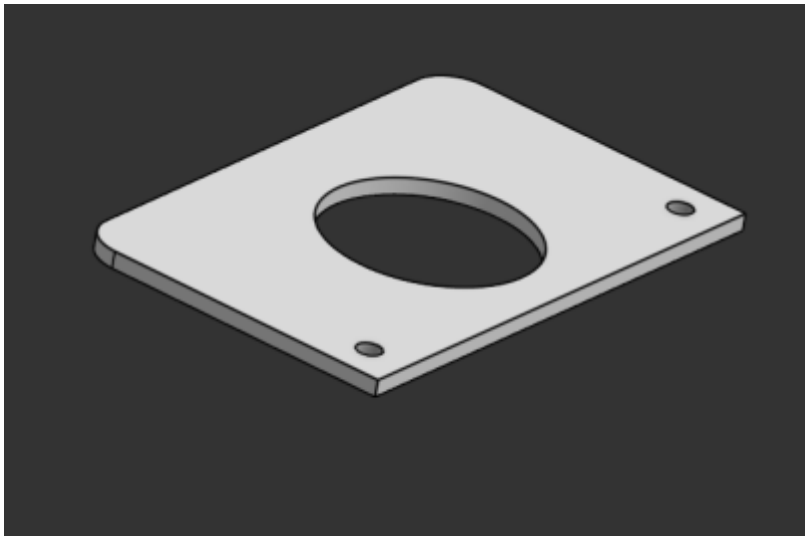


Figure 3: Picture of the roof and floor

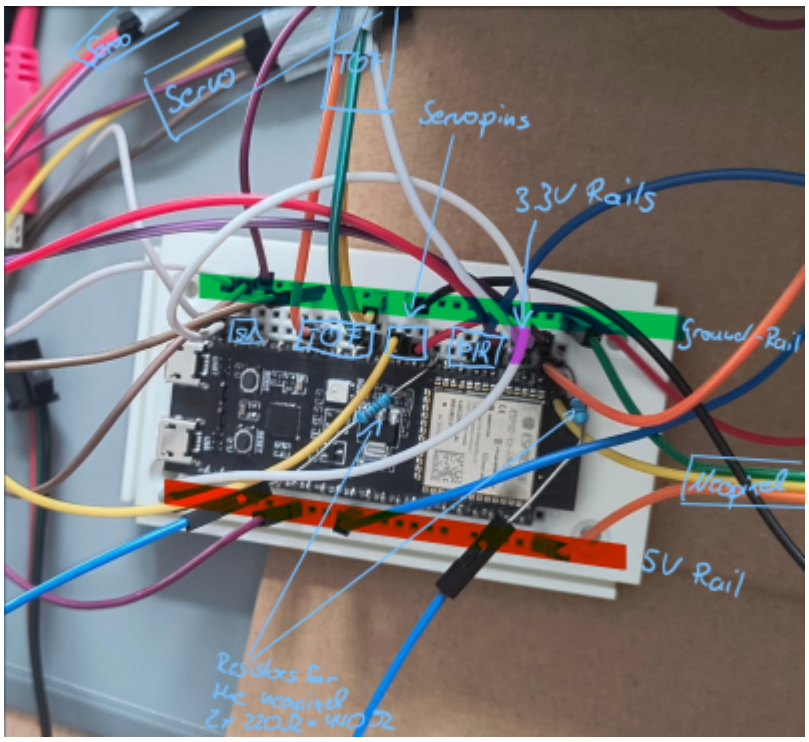
Hardware Components

by Hendrik Sensken 33359

Component	Purpose
ESP32	Main microcontroller
VL53L0X	Time-of-Flight (ToF) distance sensor
PIR sensors (2x)	Detect left/right motion
Servos (2x)	Rotate ToF and LED for tracking
NeoPixel LED	Light

Wireing

by Hendrik Sensken 33359



Libraries Used

by Hendrik Sensken 33359

Adafruit_VL53L0X: Controls the ToF sensor.

Adafruit_NeoPixel: Controls the RGB LED.

ESP32Servo: PWM servo motor control on ESP32.

Logic

by Hendrik Sensken 33359

At startup, the system performs an initial environmental benchmark by scanning from 0° to 180°, saving the distance data for each angle.

→Then it enters a loop waiting for PIR motion detection.

When motion is detected:

→It determines direction (left or right PIR).

→Performs a scanning sweep (with ToF) over the affected region.

If a significant difference from the benchmark is found:

→It activates a white LED.

→It moves a secondary servo to point an LED at the change.

The system re-benchmarks the environment every 5 minutes

Code

by Hendrik Sensken 33359

```
//--- include libaries ---
#include <Adafruit_VL53L0X.h>
#include <Adafruit_NeoPixel.h>
#include <ESP32Servo.h>


// --- Defines Pins ---
#define PIR_LEFT_PIN   4
#define PIR_RIGHT_PIN  5
#define LED_PIN        15
#define TOF_SERVO_PIN  17
#define LED_SERVO_PIN  16

// --- Constants ---
#define NUM_PIXELS         1
#define MAX_ANGLE          180
#define STEP_ANGLE         1
#define CHANGE_THRESHOLD   100     // mm
#define BENCHMARK_INTERVAL 300000  // 5 min in ms
#define PIR_DEBOUNCE_TIME  1000    // ms

// --- Globals ---
Adafruit_VL53L0X lox = Adafruit_VL53L0X();
Adafruit_NeoPixel pixels(NUM_PIXELS, LED_PIN, NEO_GRB + NEO_KHZ800);

Servo tofServo;
Servo ledServo;

int benchmark[MAX_ANGLE + 1];
bool scanning = false;
bool benchmarkReady = false;
int lastMotionDirection = -1; // 0 = left, 1 = right
unsigned long lastBenchmarkTime = 0;
unsigned long lastPirTime = 0;

// --- Function Prototypes ---
void lightUp(bool on);
void moveTofServo(int angle);
void moveLedServo(int angle);
void performBenchmark();
bool scanMotion(int startAngle, int endAngle);


//Initializes:
```

```
//   Serial monitor for debugging.
//   PIR pins.
//   ToF sensor in continuous mode.
//   NeoPixel.
//   Servos with appropriate PWM settings for ESP32.
//   Performs an initial environmental benchmark.

void setup() {
  Serial.begin(115200);

  pinMode(PIR_LEFT_PIN, INPUT);
  pinMode(PIR_RIGHT_PIN, INPUT);

  if (!lox.begin()) {
    Serial.println(F("Failed to start VL53L0X!"));
    while (1);
  }
  lox.startRangeContinuous();



  // Servos
  ESP32PWM::allocateTimer(0);
  ESP32PWM::allocateTimer(1);
  ESP32PWM::allocateTimer(2);
  ESP32PWM::allocateTimer(3);
  tofServo.setPeriodHertz(50);
  ledServo.setPeriodHertz(50);
  tofServo.attach(TOF_SERVO_PIN, 500, 2500);
  ledServo.attach(LED_SERVO_PIN, 500, 2500);

  pixels.begin();
  pixels.setBrightness(100);
  pixels.setPixelColor(0, pixels.Color(0, 0, 0)); // Ensure off at start
  pixels.show();

  performBenchmark(); // Initial benchmark
}

void loop() {
  unsigned long now = millis();

  // Re-benchmark every 5 minutes
  if ((now - lastBenchmarkTime > BENCHMARK_INTERVAL) && !scanning) {
    performBenchmark();
  }

  // Motion detection with filtering
  //Checks if PIRs are triggered over 300ms to reduce false positives.
Triggers scanning only if a direction is confidently detected.
```

```
  if (!scanning && benchmarkReady) {
    if ((now - lastPirTime) > PIR_DEBOUNCE_TIME) {
      bool leftTriggered = false;
      bool rightTriggered = false;

      // Read the PIR pins multiple times over ~300ms
      for (int i = 0; i < 6; i++) {
        if (digitalRead(PIR_LEFT_PIN) == HIGH) leftTriggered = true;
        if (digitalRead(PIR_RIGHT_PIN) == HIGH) rightTriggered = true;
        delay(50); // 6 × 50ms = 300ms total sampling
      }

      if (leftTriggered) {
        Serial.println("PIR LEFT triggered (filtered)");
        lastMotionDirection = 0;
        scanning = true;
        lastPirTime = now;
      } else if (rightTriggered) {
        Serial.println("PIR RIGHT triggered (filtered)");
        lastMotionDirection = 1;
        scanning = true;
        lastPirTime = now;
      }
    }
  }

  // Start scanning Sweeps back and forth. Scans until it finds two
consecutive sweeps with no changes.
  if (scanning) {
    int noChangeSweeps = 0;
    int start = (lastMotionDirection == 0) ? 0 : 90;
    int end = MAX_ANGLE;

    while (noChangeSweeps < 2) {
      Serial.printf("TOF Sweep (no-change count: %d)\n", noChangeSweeps);
      bool changed = scanMotion(start, end);
      if (changed) {
        noChangeSweeps = 0;
      } else {
        noChangeSweeps++;
      }

      // Reverse sweep direction for next run
      int temp = start;
      start = end;
      end = temp;
    }

    Serial.println("No more movement. Scan finished.");
    lightUp(false);
    scanning = false;
```

```
  }
}
// Sweeps 0°–180°, storing the baseline distance values from ToF.Retries if
distance is 0 (sensor failed)
void performBenchmark() {
  Serial.println("Benchmarking environment...");
  for (int angle = 0; angle <= MAX_ANGLE; angle += STEP_ANGLE) {
    moveTofServo(angle);
    delay(15);
    int dist = lox.readRange();
    if (dist == 0) {
      angle -= STEP_ANGLE;
      continue; // Retry this angle
    }
    benchmark[angle] = dist;
    Serial.printf("Angle %d° = %d mm\n", angle, dist);
  }
  benchmarkReady = true;
  lastBenchmarkTime = millis();
  Serial.println("Benchmark complete.");
}
//Reads distance at each angle.
//Compares it with the benchmark value.
//If difference > CHANGE_THRESHOLD (100 mm): Activates LED
//Moves LED servo to angle (inverted)

bool scanMotion(int startAngle, int endAngle) {
  bool foundChange = false;
  int step = (startAngle < endAngle) ? STEP_ANGLE : -STEP_ANGLE;

  for (int angle = startAngle; angle != endAngle + step; angle += step) {
    moveTofServo(angle);
    delay(15);
    int distance = lox.readRange();
    if (distance == 0) continue;

    int diff = abs(distance - benchmark[angle]);
    if (diff > CHANGE_THRESHOLD) {
      Serial.printf("Change at %d° → %d mm (was %d mm)\n", angle, distance,
benchmark[angle]);
      moveLedServo(angle);
      lightUp(true);
      foundChange = true;
    }
  }

  if (!foundChange) {
    lightUp(false);
  }
```

```
  return foundChange;
}

void moveTofServo(int angle) {
  tofServo.write(angle);
}

void moveLedServo(int angle) {
  ledServo.write(160 - angle); // Inverted to match orientation
}

void lightUp(bool on) {
  if (on) {
    pixels.setPixelColor(0, pixels.Color(255, 255, 255)); // Pure white
  } else {
    pixels.setPixelColor(0, pixels.Color(0, 0, 0));
  }
  pixels.show();
}
```

Discussion

by Berkay Kiris 34862

The smart sight light project successfully demonstrates the integration of motion detection and directional lighting to create a responsive and energy-efficient lighting system.
The core idea of the project is to provide illumination only when and where it is needed. This is achieved through the use of sensors that detect the presence of a person and activate the light only in the area immediately in front of them. As the person moves, the lighting follows, ensuring continuous visibility while minimizing unnecessary illumination.
This targeted approach to lighting offers several clear advantages. First, it significantly reduces light pollution, which is an increasing concern in both urban and residential environments. By avoiding the illumination of unused areas, the system ensures that the surrounding environment remains dark, preserving natural night conditions for humans, wildlife, and also astronomy. Second, the system leads to lower energy consumption, as the light remains off when no one is present and only lights a limited area during operation. This makes our smart sight light especially relevant in the context of sustainability and environmental impact.
Throughout the development process, certain challenges were encountered, such as ensuring reliable motion tracking and achieving smooth directional control of the light. Accurate detection is crucial to avoid unnecessary activation or failure to illuminate when needed. Future iterations could benefit from using more advanced sensors or incorporating machine learning algorithms to improve movement prediction and tracking performance.
Additionally, while the current prototype functions effectively for small indoor environments, the concept has broader applications. For example this could be used in street lighting systems or in huge industrial facilities, where motion-based, directional illumination could substantially reduce energy use while still providing safe navigation for workers and vehicles.

Concept, 3D-Modeling, code and demonstration

by Berkay Kiris 34862, Moritz von der Brake 31698 and Hendrik Sensken 33359

smart_sight_light_amc_ss2025_group_x.mp4

Conclusion

by Berkay Kiris 34862

The development of the smart sight light system has shown how combining sensor technology with targeted lighting control can lead to practical and energy-efficient solutions. By illuminating only the path in front of a moving person, it reduces both power consumption and light pollution, while still ensuring safety and visibility for people walking in its scanning area.
Overall, the project has provided valuable insights into sensor integration, motion tracking, and system responsiveness.With continued improvement, the smart night light concept can become a scalable and impactful contribution to future lighting systems.

From:
https://student-wiki.eolab.de/ - **HSRW EOLab Students Wiki**

Permanent link:
https://student-wiki.eolab.de/doku.php?id=amc:ss2025:group-x:start&rev=1753696283

Last update: **2025/07/28 11:51**