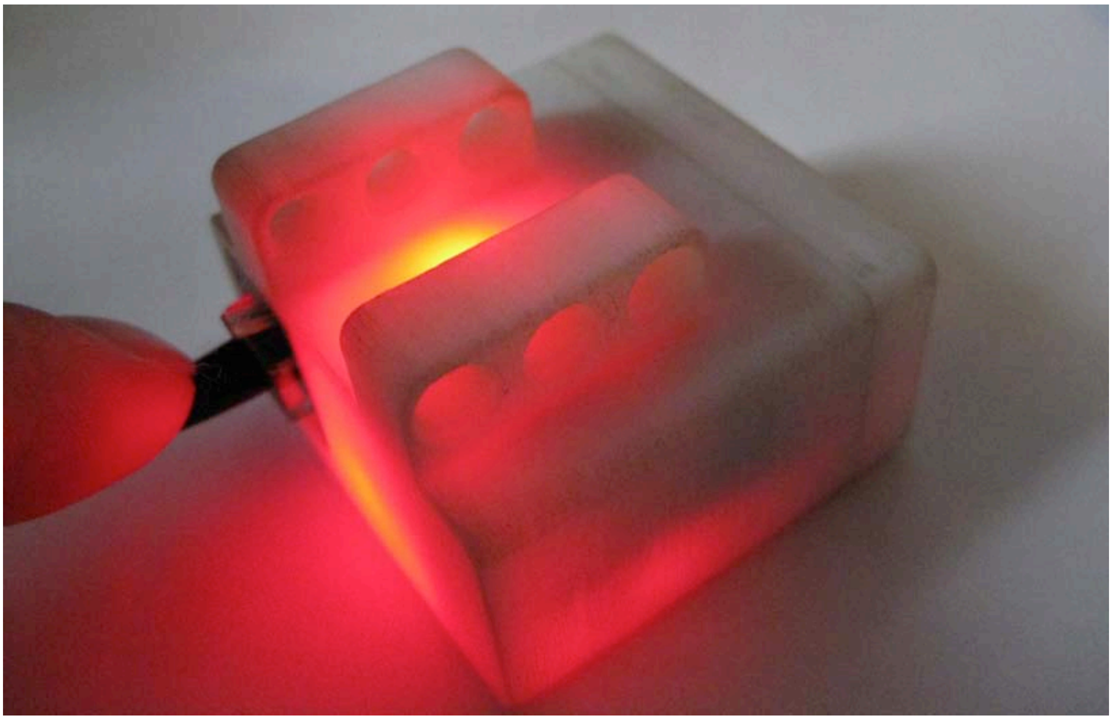


Report Technology Masterclass - Martijn Jansen

An Optical Mouse Sensor as an extension for LEGO NXT

Martijn Jansen
m.e.p.jansen@student.tue.nl
S030887

Index:

1. Introduction

Assignment Description

This project is part of the Technology Masterclass of Industrial Design at the Technical University Eindhoven. During this class, Master students get the opportunity to dive into some very diverse elements of technology like electronics, mechanics, programming intelligent robots and creating a new innovative sensor for LEGO NXT. This report describes the elaboration of the latter assignment.

Deliverables

- working model of the extension pack
- report in MS Word describing the extension pack
- technical documentation of the extension pack, such as technical drawings and UML models
- photos of the extension pack
- movie of the extension pack in action uploaded in high quality to SurfMedia and available for download from there.

2. Sensor Choice

For a robot that is able to move itself, it is essential to know in what environment it is and where in that environment it is located. Location, distance and speed are variables that can help a robot determine it's current state. A GPS-sensor is able to give these variables, but are quite expensive and not as accurate as you might need. A computer mouse is more accurate and can detect movement smaller than a millimeter. For this reason I decided to look at computer mice as a new sensor for LEGO NXT.

An 'old' computer mouse (fig. 1) detects movement mechanically through a ball inside that can rotate in any direction. There are three wheels inside that are positioned against the ball, so that the wheels are turning when the ball rolls. These wheels chop light beams in front of light sensors so a movement, speed of movement and direction can be detected. This kind of mouse is also called a ball mouse.

Fig. 1: A ball mouse[1]

A different kind of computer mouse is the optical mouse (fig. 2). The optical mouse has a sensor inside that is actually a very small camera. It detects changes in pictures of the surface underneath it. This is then converted to an amount of movement into the X- or Y-direction. A red LED illuminates the surface to increase the contrast of the pictures.

fig. 2: My optical mouse (before this project)

I chose to go further with the optical mouse sensor for the following reasons:

- it has no mechanical parts that can break or wear off;
- it can be used on more surfaces than the ball mouse;
- a ball mouse collects debris inside the casing
- it's quite small (smaller than the ball mouse);
- it's cheap;
- I see a lot of learning opportunities here.

The following chapters will describe how to create an optical mouse sensor for the NXT from a cheap optical mouse.

3. Hacking the optical mouse

I have used some tutorials I found on internet[2,3]. My approach was quite similar, but eventually I need to read the data with the LEGO NXT. My first step is to hack the mouse read the data from the sensor with Arduino[4].

Tutorial for hacking the optical mouse

This tutorial will explain what steps you need to take to hack the optical mouse and read the data with Arduino. First take an old optical mouse or buy a cheap one. Fig. 2 shows how my mouse looked like.

1. Open up the mouse

Underneath the mouse there's a screw you need to unbolt. This will open up the mouse. Here you find the circuit board with the sensor (the chip in the middle), a LED, buttons and the scroll wheel (fig. 3). Underneath the circuit board is the lens clip (fig. 4), that is used to amplify the light and functions as a lens for the sensor.

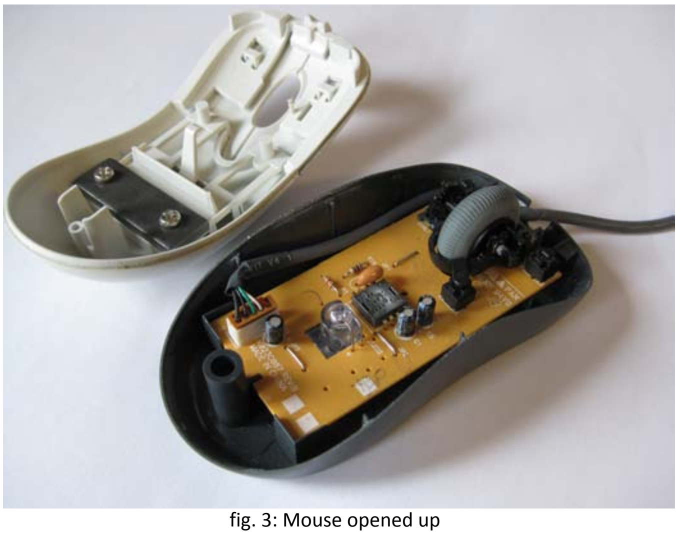


fig. 3: Mouse opened up

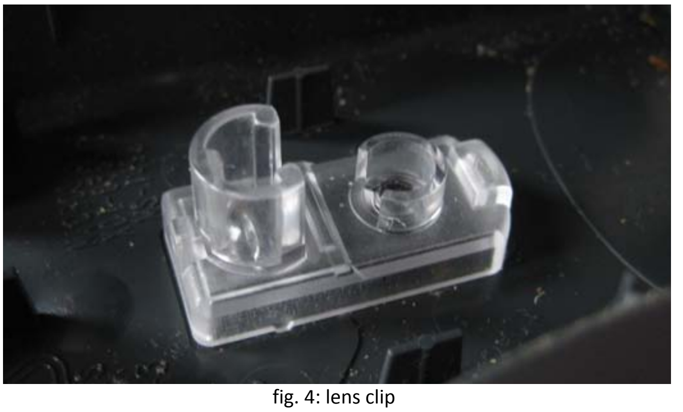


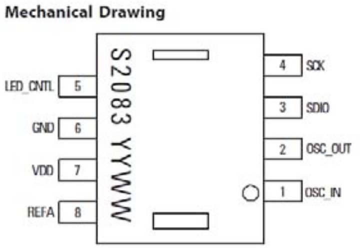
fig. 4: lens clip

2. Look up the datasheet

The sensor is located in the middle of the circuit board. Search for the datasheet of the sensor on the internet and look up the pin connections. You need the following pins:

- +5V (input voltage)
- GND (ground)
- SCK (serial port clock)
- SDIO (serial port data)

In the case of my sensor I need pin 3, 4, 6 and 7 (fig.5)



Pinout table

Pin	Pin	Description
1	OSC_IN	Oscillator Input
2	OSC_OUT	Oscillator Output
3	SDIO	Serial Port Data (input and output)
4	SCK	Serial Port Clock (Input)
5	LED_CNTL	Digital Shutter Signal Out
6	GND	System Ground
7	VDD	5V DC Input
8	REFA	Internal Reference

fig. 5 mechanical drawing + pinout table of the datasheet[5]

Make sure that your sensor doesn't have the USB-controller embedded inside the chip. The first mouse I tried to open had this kind of chip (see fig. 6), which means that you can't capture the data without a USB-protocol. Using a 2-wire serial port communication is much more preferable.

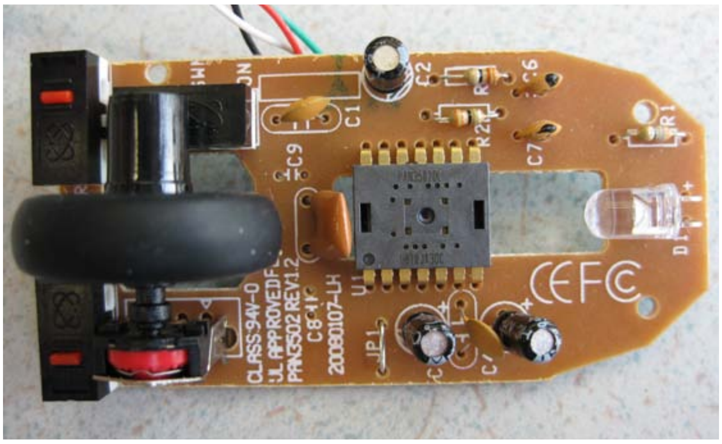


fig. 6: a mouse with the USB controller embedded in the chip of the sensor

3. Solder wires

Solder wires to the 4 needed pins. It's useful to choose different colors, so you're able to distinguish the clock and data lines easily (fig.7).

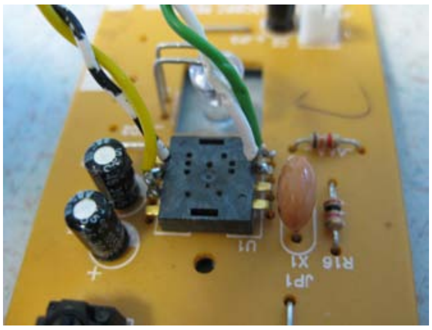


fig. 7 - Wires soldered to pins 3, 4, 6 and 7

4. Remove the USB-controller

This step is not necessary, but if you're not going to use the mouse anymore it might be better to remove the USB-controller to prevent the controller of taking over the data connection. Removing the controller can be done by removing the soldered tin and scrape the controller of the circuit board.

5. Connect it to Arduino

Choose two digital pins on the Arduino for the clock line and the data line and connect the +5V and GND of the mouse to the 5V and Gnd of Arduino. Upload the sketches (programs for Arduino) provided by Martijn Thé based on sketches of Benoît Rousseau. Make sure that the right registers If you open the Serial monitor in the Arduino software you'll see the x- and y-values. In the next paragraph I'll explain how this works.

Get the data from the optical mouse to Arduino

In the datasheet of the sensor there is a whole section about the synchronous serial port. This port is used to set and read parameters from the sensor. The port uses two wires and is a half duplex port, which means that it can talk both ways, but not at the same time. The microcontroller, in my case the Arduino, always initiates communication and is called the master (the sensor is the slave). The two wires are the serial port clock (SCK) and the data line (SDIO). The serial port clock is generated by the microcontroller.

The microcontroller can try to read parameters like the X- and Y-movement, but can also write to the sensor to change parameters like for example the configuration of the sensor. For thiswe need a read and a write operation. The clock line is generated by the microcontroller and is preferably pulsing at a length of 250 ns per pulse. The function of the clock is to synchronize data transfers.

Write operation
A write operation is initiated by the microcontroller and consists of two bytes. The first byte contains a 7-bit address and has a “1” as its most significant bit (MSB). This 1 means that the microcontroller wants to write data and the address is a register where you want to write to. The second byte contains the data you want to write. The clock synchronizes the data transfer, because the microcontroller changes SDIO on falling edges of the clock and the sensor reads SDIO on rising edges of the clock.

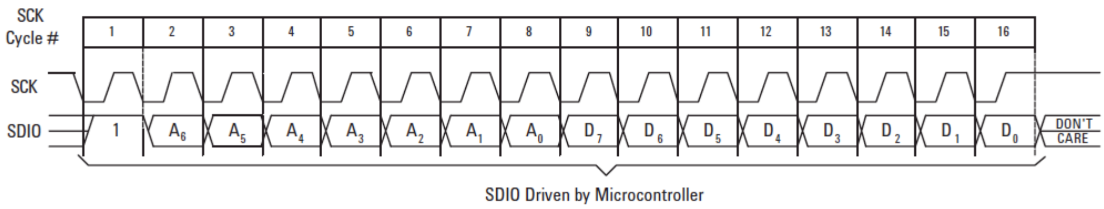


fig. 8: Write operation

Read Operation
A write operation works kind of the same; it also consists of two bytes. The first byte has a “0” as MSB (to indicate a read operation) and a seven bit address to request the data from a certain register. The second byte is the data send from the sensor to the microcontroller. After the last bit of the address there is a delay of 100 us for the sensor to prepare the data.

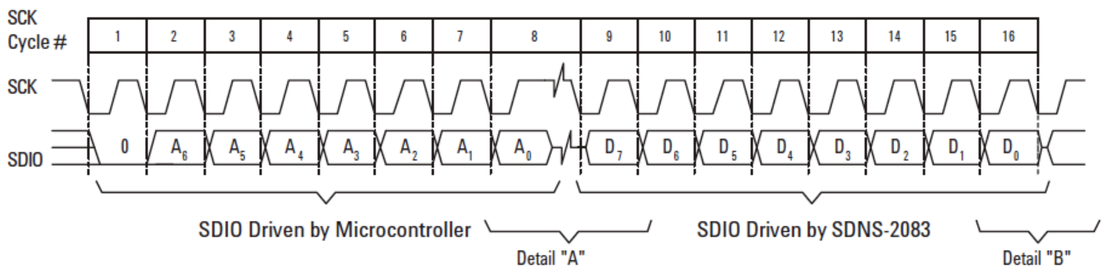


fig. 9: Read Operation

Registers
In the datasheet there is a list of addresses (fig. 10). From address 0x42 and 0x43 you can get the Delta_Y and Delta_X (X- and Y-movement). Let me give an example: If I want to read the Y movement I have to perform a read operation. The first byte the microcontroller sends is 0x42, but with a “0” as MSB bit. In bit-form this looks like this: ”01000010”. Then there is a delay of 100 microseconds and after that the sensor will give a byte containing the movement of the sensor in the Y direction.

Address	Register	Notes
0x40	Configuration	Reset, Power Down, LED mode, Forced Awake, etc
0x41	Status	Product ID, Mouse state of Asleep or Awake
0x42	Delta_Y	Y Movement
0x43	Delta_X	X Movement
0x44	SQUAL	Measure of the number of features visible by the sensor
0x45	Maximum_Pixel	
0x46	Minimum_Pixel	
0x47	Pixel_Sum	
0x48	Pixel Data	Actual picture of sensor
0x49	Shutter High	
0x4A	Shutter Low	
0x4B	Frame Period	

fig. 10: Addresses of the registers of the ADNS-2083

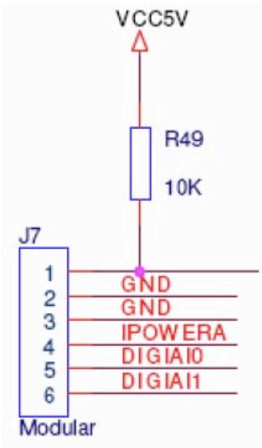
In the next chapter I'll explain how to get this data to the LEGO NXT.

4. Hacking LEGO NXT

The NXT is actually a microcontroller like Arduino. For this assignment we need to program the NXT with leJOS, a JAVA-based language for the NXT. I couldn't figure out how to read the sensor directly with the NXT using leJOS. Therefore I chose to to read data from the sensor with Arduino and let the NXT communicate with the Arduino.

LEGO NXT cables:

The cables of the NXT are actually slightly modified JR12 cables (telephone cables). The modification lies in the connector. Inside the cable are 6 wires that connect to an input port of the NXT (fig. 12).



- Pin 1, ANA — Analog input and possible current output signal
- Pin 2, GND — Ground signal
- Pin 3, GND — Ground signal
- Pin 4, IPOWERA — 4.3 Volt output supply
- Pin 5, DIGIAI0 — Digital I/O pin connected to the ARM7 processor
- Pin 6, DIGIAI1 — Digital I/O pin connected to the ARM7 processor

fig.12: Input port of the NXT[6]

Communication between Lego NXT and Arduino through I^2C:

I^2C is a protocol developed by Philips. The protocol of reading the optical mouse sensor is quite similar to this, but not entirely. In the case of the optical mouse sensor, there is no device address. You need this to start up the whole communication. For this reason I couldn't read the sensor directly from the NXT.

In my project, the Arduino is the slave and the NXT is the master. In Arduino there's a library called "Wire.h" that can be used for I^2C. With this library you can set the device address, I used the number 127. Whenever the NXT requests data, I use this library to send an array of two bytes; the x- and y-value.

In leJOS I request data by using the I2CSensor class. I created a class for the optical mouse sensor that extends the I2CSensor class. Now I can make several objects of this class, which can be useful when I want to use more than one of these sensors at the same time.
After the NXT has initialized communication with device address 127, it can request data. After this request I get a response of two bytes, which I store in an array with two indexes . The first index of the array is the x-value and the second is the y-value.

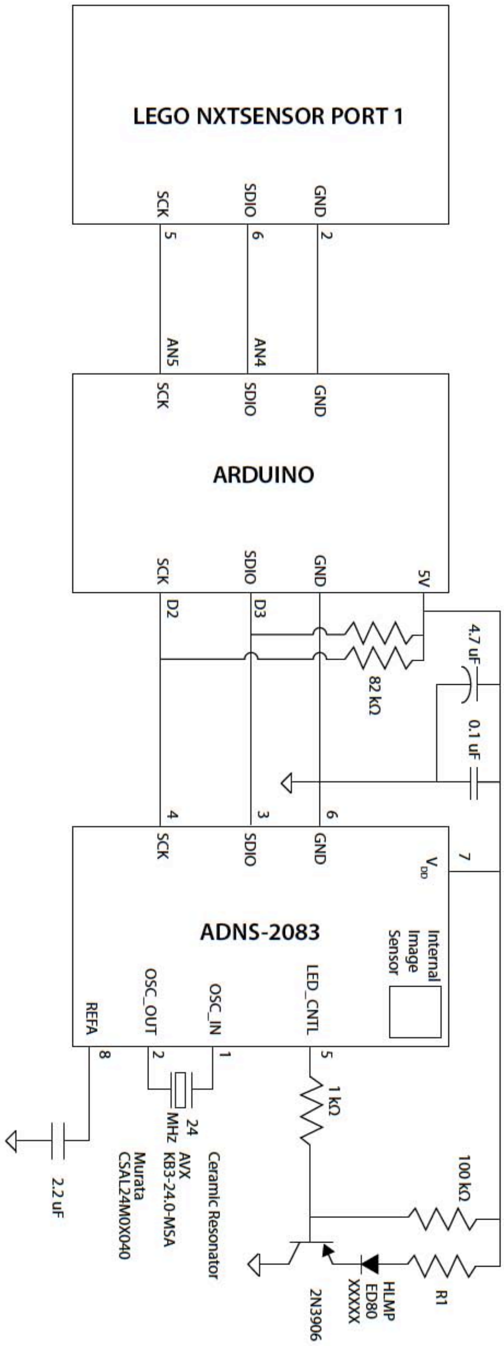
5. Implementation

This chapter describes the implementation into a working prototype.

Circuit block diagram

The circuit block diagram (fig.13) is largely based on the circuit block diagram of the ADNS-2083. By removing the unnecessary parts like the USB controller and the buttons you'll have a block diagram of the sensor with the LED. Connect the four pins (+5V, GND, SCK and SDIO) of the sensor to the right pins on the Arduino board. Connect the I^2C pins of the Arduino board (SCK = ANALOG5, SDIO = ANALOG4) to the NXT cable. The other end of the cable can be inserted into any sensor port of the NXT, except sensor port 4 (is not capable of I^2C). I used sensor port 1.

Fig. 13: Circuit block diagram for reading the sensor in NXT (rotate your page)



Mechanical Drawing

I wanted to make a sensor has almost the same form as the existing NXT sensors. The first drawings (fig. 14) give a good impression of the dimensions of the outer body.

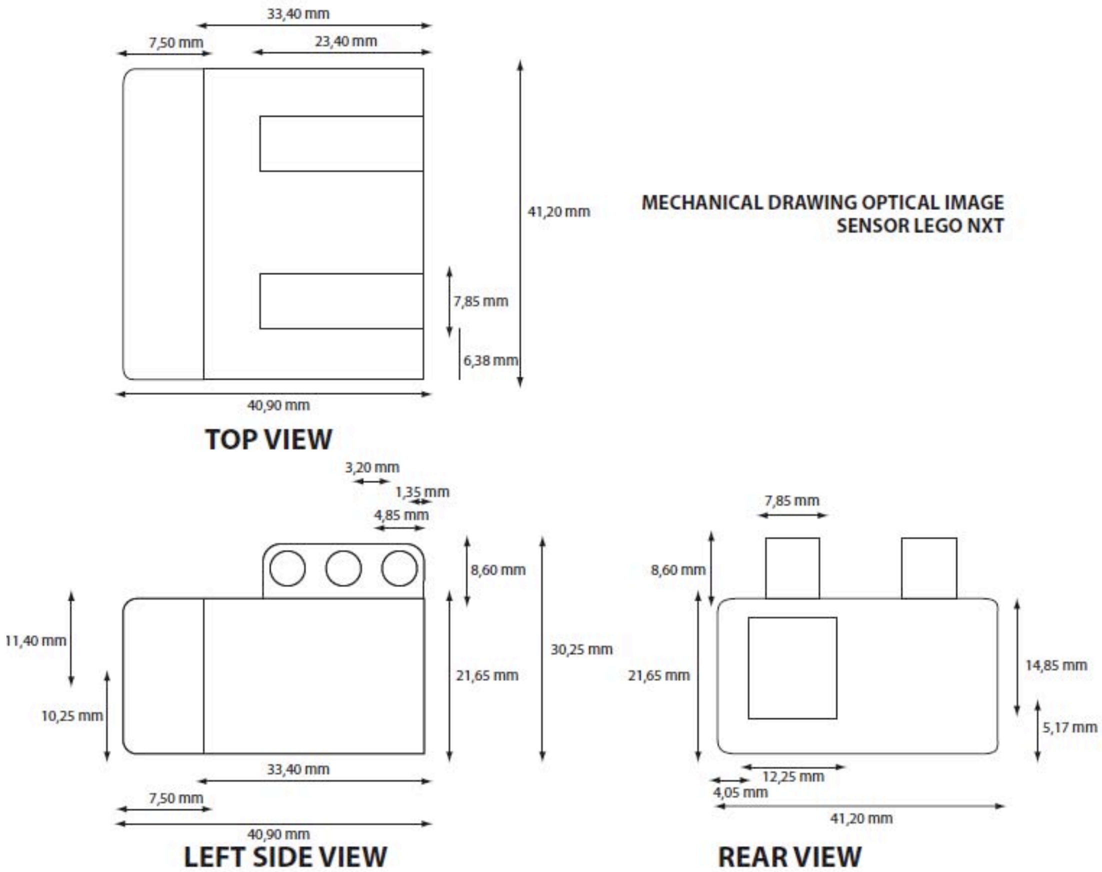


fig. 14: First impression of the dimensions

After I made this I started to create a 3D-model for 3D printing. The thickness of the material is 2mm. Inside the casing a different construction was needed. The circuit board had to be exactly 7.45 mm above the surface (fig. 15). The mouse I hacked used a different lens clip than the datasheet recommended. In my design I needed to fix the lens clip by making a raised rectangle and cut out a passage for the light to go through and a passage for the lens to make pictures (just like inside the casing of the mouse, fig.16). The implementation is shown in fig.17 and the eventual model in fig. 18.

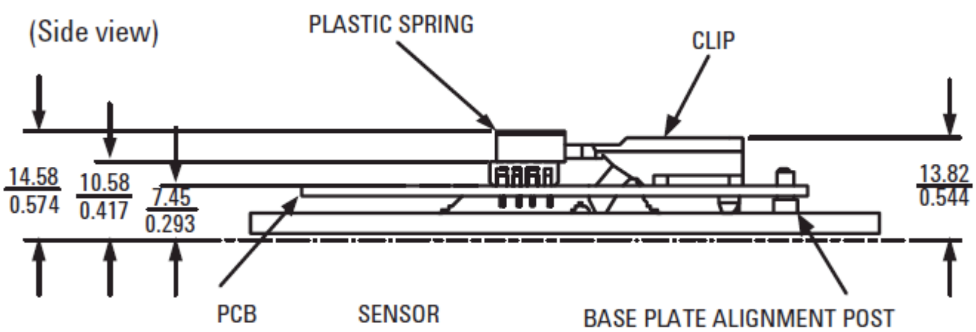


fig. 15: Assembly drawing of circuit board in side view

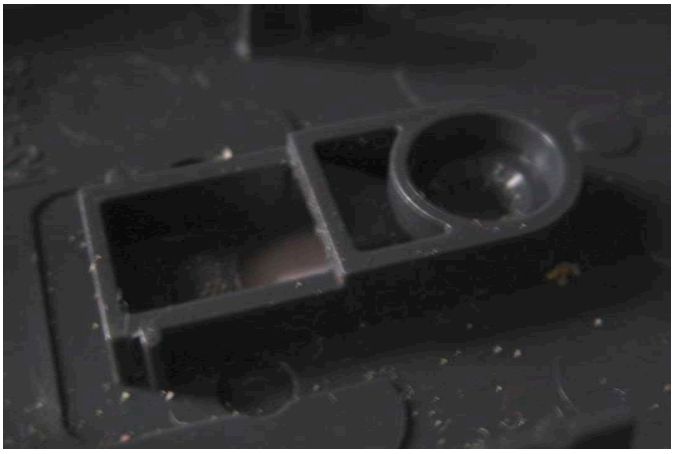


fig. 16: Lens clip holder in the mouse casing

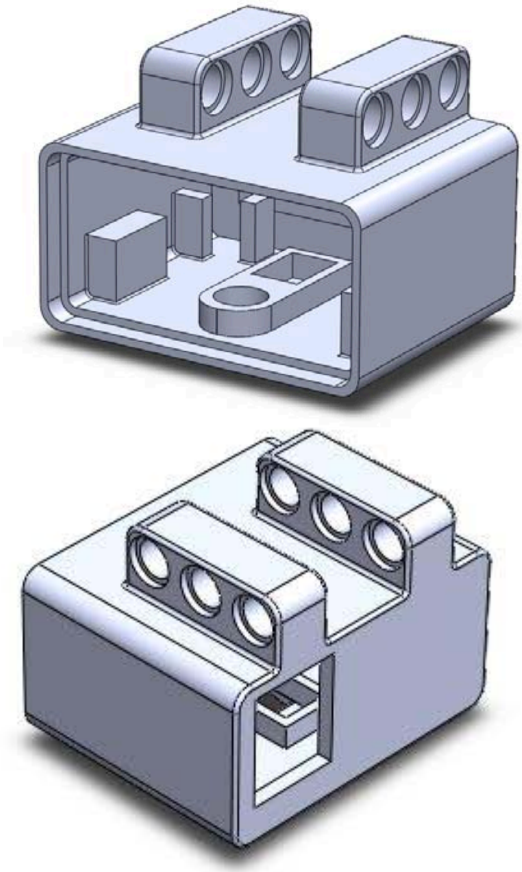


fig. 17: Two screenshots of the 3D-CAD model made in SolidWorks

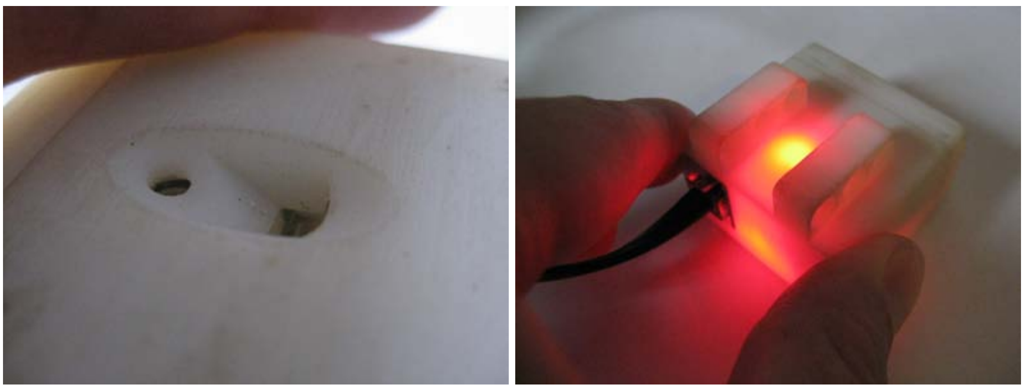


fig. 18: The prototype 3D-printed with the technology inside

Next steps

I’ve tried out the sensor on different surfaces and it works very well on white or bright surfaces. On dark or black surfaces it has more trouble to switch on. There is a setting of the mouse sensor that switches the LED to less bright when it isn’t moved for more than a second. When the sensor detects movement, the LED will get brighter again. This is a setting to save energy. The problem on dark surfaces is that the sensor doesn’t detect the first movement very well, so the LED will not emit at full power. In case of emitting with full power the sensor works fine again. I would recommend when the sensor is used on dark surfaces to change the configuration of the sensor and let the LED always on with full power.

There are also some minor improvements that can be made to the design of the casing. The lens clip only fits when it is put in with a lot of force. Trying a lit bigger margin would be better.

A smaller circuit board can be designed with the technical drawing provided in this report. This means that the sensor would decrease in size and might even decrease its broadth twice.

6. Possible Applications

Some examples of applications using the NXT optical mouse sensor are:

Measure movement
A robot can measure how far it has driven yet.

Measure velocity
For robots that have to go with a certain speed you can divide the movement with the time and you get velocity

Feedback loop for driving straight
Because the sensor measures X and Y values it can also give feedback about movement sideways. This is unwanted for example because on motor is stronger than the other. Because you detect the movement sideways you can feed this information back to the motors (just like the potentiometer in a servo motor).

Driving to coordinates
A robot can drive to an exact point in the environment, when it keeps track of the movement so far and the bends it has made.

7. References

1. Picture from: http://upload.wikimedia.org/wikipedia/commons/2/2c/Ball_Mouse_semi-inside_view.jpg
2. http://www.gurulib.com/_project/optical_mouse_hack_files/optical_mouse_hack.htm
3. http://www.martijnthe.nl/optimouse/
4. http://www.arduino.cc/
5. Datasheet ADNS-2083 (SDNS-2083) from Avago technologies
6. Lego Mindstorms NXT Hardware Delopment Kit

8. Appendix

A. Datasheet ADNS-2083

B. Lego Mindstorms NXT Hardware Delopment Kit

C. Arduino Sketch

D. leJOS classes

E. Video of the working prototype

Because Appendix C & D contain too many pages to print and are more useful to have digital, I have made them available online. Appendix E is a video and can be downloaded.

C. http://www.kansloosdesign.nl/portfolio/wp-content/uploads/2008/12/coordinates_in_mm_i2c.zip

D. http://www.kansloosdesign.nl/portfolio/wp-content/uploads/2008/12/lejos-optical-mouse-sensor.zip

E. http://www.kansloosdesign.nl/portfolio/wp-content/uploads/2008/12/filmpje.m4v