1 Requirement specification

1.1 System definition

The system is a computerized sorting system which sort out defective items during production of transparent blocks. It must relieve human operators of the defective item selection during operation. The system also controls the conveyer belt where the items are placed before sorting.

Drawing here!!

1.1.1 Requirements

1. The system shall sort out non-transparent blocks.
2. The system shall not be slower than a human operator performing the same task.
3. The system shall have a start/stop button to start and stop the conveyer belt.
4. When the conveyer belt is started the system shall be fully functional.
5. The system shall be implemented using the LEGO NXT processor.
6. The system shall be implemented using available sensors and actuators for the LEGO NXT processor.
7. A demonstration prototype must be realizable within a short time period.
8. The system shall be implemented using COTS items to the greatest extent possible.

1.1.2 Problem Domain Structure

1.1.2.1 The 3-layered model of the system

1.1.3 Application Domain Structure

1.1.4 Acceptance Test Specification

2 Implementation

2.1 Investigation

2.1.1 The Touch sensor

The Touch sensor is used to start and stop the conveyer belt, and is the simplest type of the sensor available.

It will, when activated, allow or prevent electrical current to flow through it when it's activated; a normally open activation sensor or switch will allow current when activated and prevent current when not activated. A normally closed switch has the opposite function.

A simple monitoring strategy is to periodically poll or sample the switch's state; on / off.

Two things need to be considered when polling a switch:

- The polling frequency needs to be determined so the system has an acceptable responsiveness.
- A strategy for handling noise/prell on the switch output.

The handling of prell can be implemented in software, by having a relatively high sampling rate and only consider a change in the switch's state, when having detected the same value for e.g. 5-10 samples.

Another approach is to use dedicated HW to handle the prell, e.g. by incorporate a Schmitt-trigger circuit in the switch.

The trade off here is the need for higher sampling rates vs. a higher price for the switch.

2.1.1.1 Analysis

The LEGO Touch sensor comes with a Schmitt-trigger circuit; hence the software needn't consider prell.

Using the lejOS it's possible to setup an event listener to get the sensors state as a call-back in the application without using polling from the application.

The analysis consists of a small program which shall determine if the lejOS supported event listener is sufficient to get good responsiveness or if it is necessary to implement a polling scheme from the application.

2.1.1.2 Program

2.1.1.2.1 EntryPoint Class (Main)

```
package gis.group3.touchsensor.analysis;
import lejos.nxt.Button;
import lejos.nxt.SensorPort;
import lejos.nxt.TouchSensor;

public class EntryPoint {
	public static void main(String[] args) throws InterruptedException {
		TouchSensor ts = new TouchSensor(SensorPort.S1);
		SensorPort.S1.addSensorPortListener(new TouchSensorListener(ts));
		Button.ESCAPE.waitForPressAndRelease();
	}
}
```

2.1.1.2.2 TouchSensorListener Class

```
package gis.group3.touchsensor.analysis;
import lejos.nxt.LCD;
import lejos.nxt.SensorPort;
import lejos.nxt.SensorPortListener;
import lejos.nxt.TouchSensor;

public class TouchSensorListener implements SensorPortListener
{
	public TouchSensorListener(TouchSensor ts)
	{
		mts = ts;
	}
	public void stateChanged(SensorPort port, int oldValue, int newValue){

		if (mts.isPressed())
		{
			LCD.drawString("Touch pressed", 1, 2);
		}
```

```
                else
                {
                        LCD.clear();
                }
                LCD.refresh();
        }
        TouchSensor mts;
}
```

2.1.1.3 Conclusion

The program shows that the lejOS support for Touch sensor using event listeners is in fact highly responsive and the conclusion is therefore that the event listener can be used for the application.

2.1.2 The conveyer belt motor

A simple motor can be used for the belt drive since there are no requirements specifying that the belt must have adjustable speed. An A/C motor would be a simple and inexpensive choice for the application.

However, for the prototype the standard LEGO Servo motor is selected. The Servo motor is a DC motor with a Tacho sensor which is a flexible and widely used motor in many applications.

2.1.2.1 Analysis

As described, there are no requirements specifying that the belt must have adjustable speed; hence the motor is simply used at maximum speed. If a speed control were to be added, the power vs. speed ratio should be investigated, using the motors build-in taco meter.

2.1.2.2 Conclusion

The motor is used at maximum speed since no speed control is needed. In order to control the motor, using the Touch sensor, the previous program used for analysis of the Touch sensor is changed to start/stop the motor instead of writing to the LCD. However the Touch sensor shall be used as a toggle switch; the first activation starts the motor, the second stops the motor.

2.1.3 LightSensor

There are many sensors that may be used to determine the transparency and/or colour of an object, yet within the standard sensor selection of LEGO Mindstorm® the LightSensor is an obvious choice. Choosing a standard component generally has the following advantages and disadvantages.

Advantages

- Readily available.
- Tried and proven.
- Support and API already implemented.
- Cheap in low numbers

Disadvantages

- General purpose, meaning designed for more that just for transparency and colour detection. This means that the LightSensor may not be as efficient or accurate as if it was specifically designed for the purpose.
- Expensive in high numbers.

For this assignment the standard LightSensor should be sufficient, yet to be sure a series of tests and analysis should be performed.

2.1.3.1 Characteristics

The LEGO LightSensor can operated in two modes; ambient and reflective. Ambient means that it measures the light level using simple passive measurements. This form of sensor is often used to detect the LUX-count in a room, indicating the room's adequacy for office space with respect to light. The reflective mode is an active mode, where a flood-light (red LED) is activated and the reflected light is measured.

It would be possible to use the ambient mode to detect transparency, if a stationary light-source was placed across form the sensor and the items to measure be transported between the light source and the sensor.

An alternative is to use the reflective mode to register how much light is reflected of the item, thereby measuring transparency.

Both solutions will be sensitive to changes in the surrounding light-level, as normal light ("white" light), is a broad spectrum light containing elements of all colours, so there is no way to filter out specific frequencies, even if the sensor allowed such filtering, which it does not.

There are alternative ways to solve this. Minimizing changes in surrounding light-level is obviously a good solution. This may be done by shielding the sensor and item being measured (using a fixed, stationary light source inside the shielding to generate some light may be needed, and is mandatory in the ambient mode). This may be augmented by continuous calibration of the sensor, thereby filtering out slow changes in the surrounding light-level. A final solution is to simply place the entire setup in a room with a fixed stationary light-source and with no uncontrollable light-sources or shadow-casters (e.g. windows and people). What level of shielding and continuous calibration is required should be the subject of further analysis.

The way the LightSensor "sees" in reflective mode is by detecting how much light is reflected by the object in front of the sensor. The flood-light source is a red LED operating in the infrared spectrum, just under visible light, yet unfortunately the sensor is not able to only operate in this spectrum, and the entire spectrum is detected by the sensor. Had the sensor been configurable to only receive the infra-red reflections it would not have been a total solution, as normal "white" light contains all frequencies, incl. infra-red. In order for the sensor to eliminate external light-sources the sensor must be calibrated, thereby allowing it to ignore the general light in the room, as long as that light does not change. What the sensor "sees" is an integer which, in normalized form, range from 0 to 1023. This value may be translated to a form of grey-scale representation of the "colour" of the item in front of the sensor. Unfortunately this does not match well with transparency, as a transparent object will be indistinguishable from a black object, as neither will reflect any light (ideally transparent and ideally black). Placing an external light source across from the sensor may solve this problem, or perhaps the assignment should be altered to sort by colour not transparency??? The accuracy of the LightSensor at different distances from the object, and in different external light levels should be the subject of further analysis.

Knowing that the external light sources may change also means that faulty readings may occur until a new calibration can be performed. An important step to avoiding faulty behaviour is to detect these readings as faulty before any action is taken. One way of filtering out bad reading is to take multiple reading before making a decision, which allows for a single faulty reading being detected and ignored. For this to be possible the sampling and detection of the LightSensor must be considerably faster than the movement of the conveyor belt. Assuming the conveyor belt is moving a 1m/s and the object is 5cm long, and we wish to have 10 samples within the object, we must perform sampling at 200Hz, as 5cm is covered in 50ms and 10 samples in 20ms requires one sample per 2ms, or 200Hz. The equation is Hz = (<conveyor speed>[m/s] / <object size>[m]) * <sample count>. It may be possible to sample at 200Hz, but this does not mean that the sensor is able to function at this speed. It depends on its reactivity - how long it takes a change in the external condition to manifest in the analogue reading. If the sensor is sluggish, meaning that a light-change has to be present for e.g. 10ms before the analogue value is updated, it will not be possible to operate the sensor at 200Hz. The reaction time of the light sensor along with the possible sampling speed must be the subject of further analysis.

We may also need to recalibrate when the light in the room changes. This could be done by determining that the sensor is between items and then comparing the reading to the calibration value. If it differs it is time for a recalibration. Whether such an on-the-fly calibration is possible must be the subject of further analysis.

Analysis required

- Reactiontime of LightSensor.
- Accuracy of LightSensor as function of distance to object
- Accuracy of LightSensor as function of external light source
- LightSensor sensitivity to external light changes.
- LightSensor sampling frequency range and main processor operating frequency
- LightSensor calibration performance.

2.1.3.2 Analysis

The analysis consists of writing a set of small test programs and creating some test-setups to shed some light (no pun intended) on the LightSensor, with respect to the issues mentioned above.

In and Table may be seen a hyphothetical test-setup, including "all" the parameters of interest.

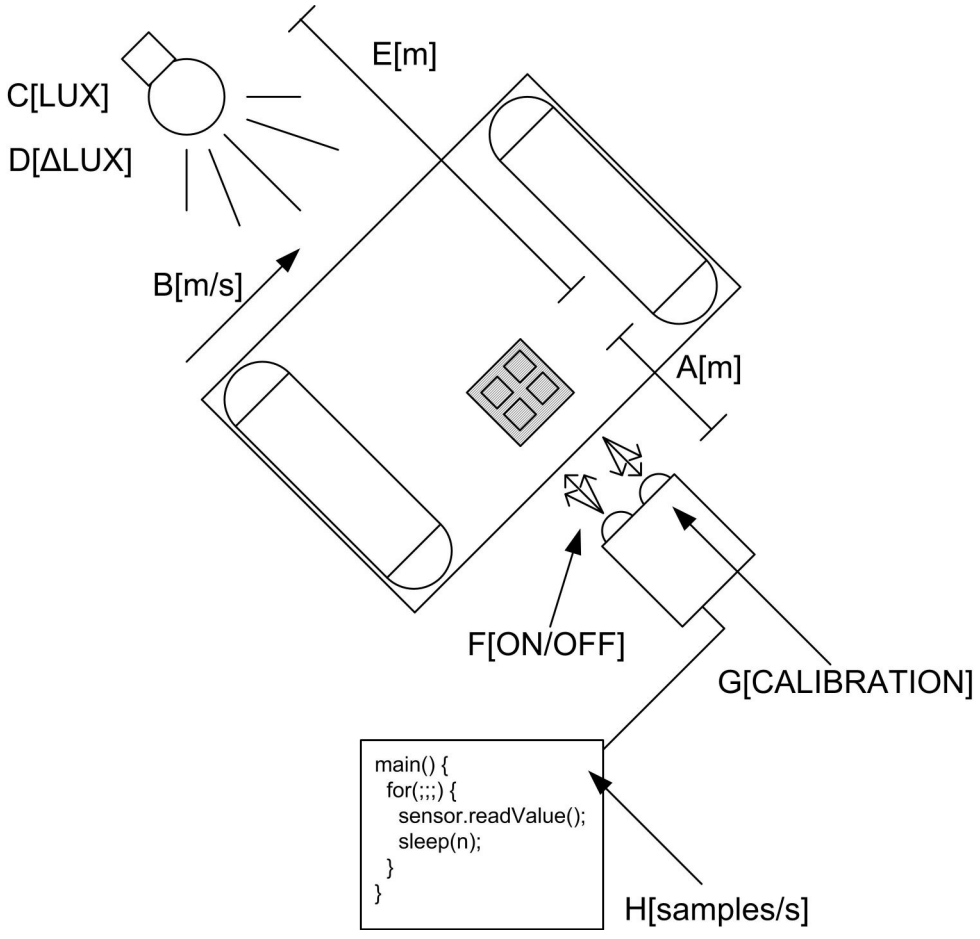

Figure 1: LightSensor analysis set-up

ID	Description
A	The distance from the LightSensor to the object it is sensing. This is important to measure in order to determine the optimal location of the LightSensor relative to the objects on the conveyer-belt, so as to achieve the most accurate readings. This may be tested by moving the LightSensor relative to the object and logging the reading-accuracy.
B	The speed of the conveyer-belt. This is very important, as the sampling frequency and the response-time of the LightSensor is directly related to the maximum speed of the conveyer-belt. To determine the response-time, a simple program samling at maximum speed may be written, and the speed of the conveyer-belt increased until sampling fails (either due to sampling frequency insufficient or response-time issues).
C	The light in the room may have profound effect on the LightSensors accuracy, and this may be tested by running identical tests, but with different external light levels. (no sampling should be done during the level change).
D	The change in lighting may also affect the accuracy of the readings. This can be determined by sampling while the lighting is being turned up or down.
E	The location of the external light source relative to the object being sensed and the LightSensor is also important. This may be measured by moving the light source while sampling.
F	Whether the flood-light of the LightSensor is on may affect the accuracy of the readings.
G	The calibration of the LightSensor will affect the readings, and testing for the best configuration is very important, as well as determining configuration drift during operation and the possibility of re-calibrating.
H	The sampling frequency is also important, as it may be that the response-time of the sensor is faster than the max sample-rate, making the sample-rate the important configuration or vice versa. It should be determined what sampling frequency can be achieved, also under the condition of servicing other peripherals.

Table 1: LightSensor analysis set-up description

As the test programs are relatively simple, the bulk of the work is in the test-setup. Naturally we do not have the necessary laboratory equipment for a proper test, e.g. LUX controlled lights, dark-rooms, strobe-lights, high-precision motors for movement-test, we will have to settle for the "good-enough" analysis.

We will run four tests, designed to shown the effects of calibration, the reaction-time vs. sampling time, the sensitivity of the sensor to external lighting and the affect of using the flood-light.

2.1.3.3 Testing calibration

1. In a room with a fixed external light place the LightSensor facing a monotone wall.
2. Run program "ConfiguraionTest".
3. Move a solid dark object in front of the sensor.
4. Move a solid light object in front of the sensor.
5. Move a semi-translucent object in front of the sensor.
6. Move a transparent object in front of the sensor.
7. Press button 1 on MindStorm control while nothing is in front of the sensor.
8. Repeat 3 - 6 (with calibration) and compare the results of the two run.

2.1.3.4 Testing reaction-time

1. In a room with a fixed external light place the LightSensor facing a monotone wall.
2. Run program "SpeedExtLightingTest".
3. Use a camera with a flash to create three single short sharp bursts of light about 30 seconds apart.
4. Move a solid dark object rapidly past the sensor. Repeat three times about 30 seconds apart.
5. Analysis read-out and determine if all three flashes and the fast motion was caught - this will indicate a high reaction-time. Sampling time can be read on the display.

2.1.3.5 Testing external light influence

1. In a room with a movable and dimmable external light place the LightSensor facing a monotone wall.
2. Run program "SpeedExtLightingTest".
3. Turn the external light up and down and move it around the room.
4. Analyse the results to determine sensibility.

2.1.3.6 Testing flood-lights

1. In a room with a fixed external light place the LightSensor facing a monotone wall.
2. Run program "FloodLightTest".
3. Move a solid dark object in front of the sensor.
4. Move a solid light object in front of the sensor.
5. Move a semi-translucent object in front of the sensor.

6. Move a transparent object in front of the sensor.
7. Press button 1 on MindStorm control while nothing is in front of the sensor.
8. Repeat 3 - 6 (flood-light and calibration) and compare the results of the two run.
9. Compare results with results of testing calibration.

Naturally this is a set of very simple tests, and we have as much as possible only modified one parameter at a time. We believe it is sufficient for this analysis.

2.1.3.7 Results

Pending.

2.1.3.8 Conclusion

2.1.4 The deflector arm actuator

The deflector arm is used to remove defective blocks off the conveyor belt. It is mounted beside the conveyor belt and is attached onto an actuator. When a defective block must be removed from the conveyor belt the deflector arm rotate 45 degrees to push/deflect the block off the conveyor belt.

When the actuator rotates the deflector arm swiftly it might not stop immediately, since both motor and deflector arm have inertia. It is necessary that rotation of the deflector arm is roughly accurate i.e. when the deflector arm swings 45 degrees we can trust it has swung 45 degrees within a margin of few degrees, say, ± 2 degrees.

2.1.4.1 Analysis

The LEGO actuator comes with a built in 9V DC motor with a tachometer sensor. LejOS provides three instances, one for each actuator, with methods for controlling speed, angle and direction etc.

The analysis will investigate how much the actuator will deviate when it rotates 45 degrees and back again. The deflector arm must be swift, hence the actuator must rotate at highest possible speed (170 rpm @ 9V). A small program rotates the actuator 45 degrees and print out the angle. This gives a good indication how much the actuator deviates from the desired angle.

2.1.4.2 Program

```
import lejos.nxt.*;

public static void main(String[] args)
{
  int count = 0;
```

```
  LCD.drawString("Inertia Test", 0, 0);

  Button.waitForPress();
  Motor.A.setSpeed(1020); // 1020 degrees/sec ~ 170rpm @ 9V

  while (Button.waitForPress() != 8) // End the program when ESCAPE is hit
  {
    Motor.A.resetTachoCount();
    Motor.A.rotate(45,false);  // block and rotate 45 degrees clockwise
    count = Motor.A.getTachoCount();
    LCD.drawInt(count,0,1);
    Button.waitForPress();

    Motor.A.resetTachoCount();
    Motor.A.rotate(-45,false); // block and rotate 45 degrees counter clockwise
    count = Motor.A.getTachoCount();
    LCD.drawInt(count,0,2);
  }
}
```

2.1.4.3 Conclusion

The analysis has showed two things:

- The actuator is able to rotate the deflector arm 45 degrees and keep deviation within a margin of ± 2 degrees.
- The methods used in the test program are able to regulate the actuator accurately.

3 Reference

http://www.cs.aau.dk/~apr/ITVHSI/

http://www.philohome.com/motors/motorcomp.htm