Sorting of objects using additional sensors

Ioana-Denisa Barbos
AXHT 3085/3
Informatik

HTBLuVA

St. Pölten, Austria
ioana-denisa.barbos@htlstp.at

Tiago Lourenço da Silva
AXHT 3085/3
Maschinenbau

HTBLuVA

St. Pölten, Austria
tiago.lourencodasilva@htlstp.at

Anna Theis
AXHT 3085/3
Maschinenbau

HTBLuVA

St. Pölten, Austria
anna.theis@htlstp.at

***Abstract*— This text is about the problems that can be faced when trying to recognize the colors correctly and how the light and color sensors work. In this robot the TCS3200 color sensor was used because of the variation of the IR light sensor. One possibility to connect the TCS3200 to the Wallaby to detect the colors red and green is via an Arduino.**

Keywords*— *color sensor, TCS3200, ESP32, Arduino, conveyor, container, programming, XHT 3085/3

I. Introduction

Many things need to be sorted. For that particular reason, it is a very regular problem. An example would be recycling, where different materials or parts must be categorized so that they can be reused or repaired. [1] Therefore, it is no surprise that our situation was similar. Because of the decision that the XHT 3085/3 was supposed to sort the poms, a system that was quick and reliable had to be developed.

II. Concept and Design

The first thing that comes to mind when thinking of a sorting system is a conveyor belt. For that reason, a prototype was designed, which consisted of the belt itself, two sensors and a divider in a drop shoot. The idea behind it was, that the collected poms would be pulled onto the conveyor belt and the sensors located on the sides would decide which color the pom had. Depending on that result, the following shoot would open either the left or the right hole. These containers would get filled up during the way to the other side of the table and let the poms fall into the newly attached transporter.

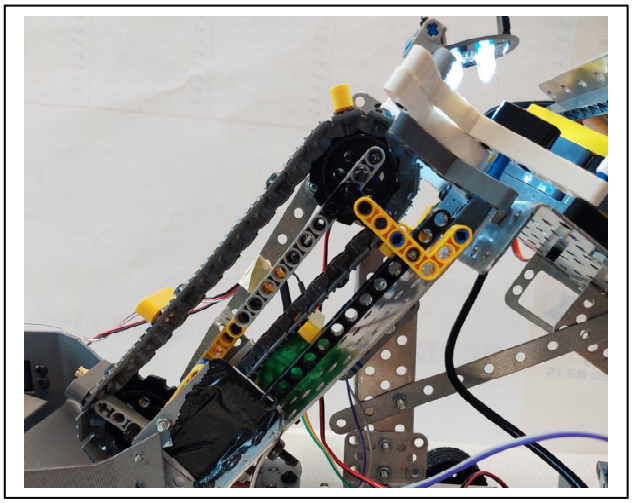

Figure 1: conveyor belt and color sensor

III. Implementation

Due to the lack of space, only one of the sensors ended up being used instead of the originally planned two, which might result in a less accurate system. Another difference is the location of the poms while transported by the conveyor belt. A result of experimentation found that the system would run better if the poms were squeezed in between a metal plate and the belt. Another positive result of this change is that the conveyor belt can have a much steeper angle, which saves a lot of space.

IR Light Sensor:
The first choice for recognizing the colors was the IR (infrared) light sensor of the botball set. The tests resulted in the following analog values [2], [3]:

distance	*1mm*	*5mm*
black	~3800	~3300
white	~160	~250 (varies greatly)
green	~190	~2400 (varies greatly)
red	~190	~1300 (varies greatly)

Table 1: Measurement of Botball IR Light Sensor

Due to the variation at a longer distance and almost identical values at a smaller distance the IR light sensor no longer was the best option.

Color Sensor:
As the light sensor was not suitable, a color sensor was due. Those photoelectric devices can emanate light and identify the color of reflected light from an object in the RGB scale with different wavelengths [4], [5], [6]. The first option for a sensor was the TCS34725 [7]. The only disadvantage was that when there is no lighting present, the values are more likely to be dependent on the ambient light. At last, the TCS3200 was used, because it has four LEDs as additional light, which is often used in projects. First the pins S2 and S3 need to be set in order to activate the photo sensors for

the detection of either red, green or blue. The more of the chosen color is seen by the sensor, the higher the output frequency is [8]. The connection to the Wallaby from TCS3200 color sensor was established via an ESP32. It is a microcontroller with integrated Wi-fi and Bluetooth. On top of that, it achieves very low power consumption through power saving features including multiple modes of operation. It works on the same voltage as the wallaby itself (3.3 Volt), which helps establishing the connection between the controllers.

First, a program for the ESP32 had to be written. In the first six lines of code the pins on the sensor were defined. The pins S0 and S1 were for the output frequencies scaling selection inputs. Pins S2 and S3 were the inputs for the photodiode color selection. So, you can select the color to be detected. There are photodiodes available for red, green and blue. The PIN_COLORSENSOR_OUT is used as an input for the frequencies, which represents the intensity of the colors. Lastly, there is the PIN_CON_WALLABY pin, which is used as an output, in order to use the program of the sensor with the existing program on the Wallaby.

```
#define PIN_S0 2
#define PIN_S1 3
#define PIN_S2 4
#define PIN_S3 5
#define PIN_COLORSENSOR_OUT 6
#define PIN_CON_WALLABY 9

unsigned long red = 0;
unsigned long green = 0;

float redFreq = 0;
float greenFreq = 0;
```

Figure 2: Code of Pindefinitions and Libraryimplementation

```
void color(void);

void setup()
{
  pinMode(PIN_S0, OUTPUT);
  pinMode(PIN_S1, OUTPUT);
  pinMode(PIN_S2, OUTPUT);
  pinMode(PIN_S3, OUTPUT);
  pinMode(PIN_COLORSENSOR_OUT, INPUT);
  pinMode(PIN_CON_WALLABY, OUTPUT);

  // Setting frequency-scaling to 2%
  digitalWrite(PIN_S0, LOW);
  digitalWrite(PIN_S1, HIGH);

  Serial.begin(9600);
}
```

Figure 3: Code of Setup

After defining some other variables, the program with defining the pins as an out- or input and setting the frequency-scaling to 2% in order to read out more defined values, can be started.

The main program – void loop():
In the beginning of the loop the color function is called. In this function the values of the color sensor can be read. In order to read red both pins S2 and S3 need to be set to low. The frequency is calculated by 1/ time period, using the time of the pulse read by the PIN_COLORSENSOR_OUT with 3.3 V positive the pulse of the PIN_COLORSENSOR_OUT with 0 V. The same is done for getting the green value except that both pins S2 and S3 are high. Another thing that was found out after experimenting, was that the value for the greenFreq (green frequency) was way too high and 70% ended up being used as a correctional factor according to the datasheet page seven[8].

```
void color() //read values of Colorsensor
{
  //read red
  digitalWrite(PIN_S2, LOW);
  digitalWrite(PIN_S3, LOW);
  red = pulseIn(PIN_COLORSENSOR_OUT, HIGH)
+ pulseIn(PIN_COLORSENSOR_OUT, LOW);
//Period in us
  redFreq = 1000000 / float(red);  //f=1/T Frequency
in Hz
  delay(50);

  //read green
  digitalWrite(PIN_S2, HIGH);
  digitalWrite(PIN_S3, HIGH);
  green = pulseIn(PIN_COLORSENSOR_OUT,
HIGH) + pulseIn(PIN_COLORSENSOR_OUT,
LOW);  //Period in us
  greenFreq = 1000000 / float(green);  //f=1/T
Frequency in Hz
  greenFreq = greenFreq / 0.7;  //Correctionfaktor
according to datasheet p.7 is 0.7
  delay(50);
}
```

Figure 4. Code of color frequency detection

After that, these values could be used to determine whether the color, which was read out is red or green. This was done by looking up which value was higher. If redFreq (red frequency) was higher, the PIN_CON_WALLABY pin was put on high – 3.3V. Otherwise the PIN_CON_WALLABY would stay on 0V. Based on whether the pin has full voltage or not, this information could be used with the main program on the wallaby using the same system as with a toggle button or any other sensor.

```
Serial.print("R= ");
  Serial.print(redFreq);//printing RED color
frequency
  Serial.print(" ");

  Serial.print("G= ");
  Serial.print(greenFreq);//printing GREEN color
frequency
  Serial.print(" ");


  if (redFreq / greenFreq > 0.8 && redFreq /
greenFreq < 1.2) {
    Serial.print("too similar");
  }

  else if (redFreq > greenFreq)
  {
    delay(250);
    Serial.print("Red detected");
    digitalWrite(PIN_CON_WALLABY, HIGH);
  }
  else if (greenFreq > redFreq)
  {
    delay(250);
    Serial.print("Green detected");
    digitalWrite(PIN_CON_WALLABY,  LOW);
  }


  Serial.println();

  //delay(2000);
```

Figure 5: Code of color determination

Establishing the connection to the Wallaby:
These values could be used in the python program uploaded on the wallaby. In order to do that, the pins must be connected to the Wallaby. First, it had to be determined where the pins had to be plugged in. This was done by using a voltmeter. The bottom most pin on the Wallaby is ground, in the middle is positive 3.3V and the pin closest to the display is for the sensor input.

The ground pin of the Wallaby must be connected to the ground pin of the ESP32. The signal pin of the Wallaby needs to be connected to the PIN_CON_WALLABY of the ESP32. The 3.3 Volt pin is not needed.

Finally, it could be seen if the program worked. This was checked in the analog sensor list. After trying to plug the pins into a digital sensor port it became obvious that this was also possible. The difference to the analog sensor pin was, that it did not read values from 0 to 3.3 V but 0 and 1. This ended up being easier to use in the already existing program. Additionally, a way of supplying the ESP32 and the sensor with power had to be found. But after a short time of experimentation, the USB port of the Wallaby seemed viable. Therefore, the same cable that was used to upload the program to the sensor could also be the power connector.

Using the sensor to sort the poms:
After that the sensor had to be installed in a proper way to read out the color of the pom properly and quickly for the following slider, which lets the poms through in only one direction – to the red or green container, to move in time and let the pom fall into the right container. Therefore, the needed position of the sensor had to be figured out. Another problem was reading out one pom at a time, which was solved, by squeezing the poms into the conveyor resulting into a proper upwards movement and only one at a time having space in the small area in between.

Connecting the Servo:
Using the servo with the Wallaby resulted in a problem, due to the color sensor and the servo having to work independently from the main code. The following experimentation connecting the Servo directly to the ESP32 caused several problems. Connecting the servo directly to the ESP32 did not work and it was not clear why. This led to several experiments using different controllers, like a mini-Arduino and lastly the Arduino MEGA. Several measurements showed that the output frequency and the voltage were correct on the pins that were used to control the servo. However, after connecting the servo, the program crashed, and the servo did not move. After measuring the Voltage during the servo being plugged in, it showed that the Voltage dropped several times. This was because of the servo, which when plugged in uses about 200 mA, which results into the Arduino not getting enough current and shutting down. It turned out that there needed to be an additional power supply connected to the servo. For this, it is possible to use the Wallaby as a power supply. The Signal cable is connected to the Arduino MEGA, as well as the ground. The 5V cable and the ground is connected to the Wallaby. Therefore, the power supply of the servo is completely separate to the Arduino Mega and does not use the power of the Arduino.

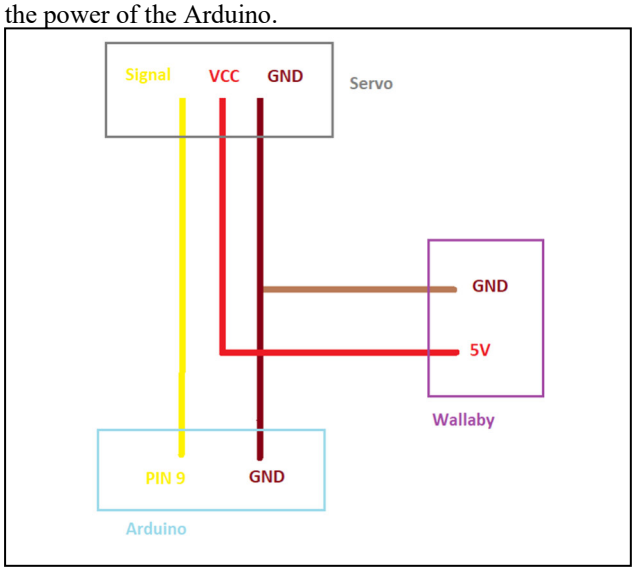

Figure 6: Cable plan of servo connection

Advantages of running the program for the color sensor separately:

- the sorting system does not impact the main program in its fluidity
- improvements to the connection between the color sensor and the servo can be done quickly because of it being in the same program

Disadvantages of running the program for the color sensor separately:

- having separate programs for different things can cause a mess
- the programs use different programming languages which can lead to maintenance problems

The Sorting Program:

To control a servo with the Arduino it is required to install the Servo library under Tools – Manage Library – search Servo. This is built-in by Arduino and allows Arduino boards to control a variety of servo motors. Therefore, the program needed to be modified.

Including the library was done with the #include <Servo.h> function. Then you need to define the pin to which you connect the signal cable of the servo. After that, an object named servoMotor is created of the type Servo by writing "Servo servoMotor;".

```
#include <Servo.h>

#define PIN_S0 2
#define PIN_S1 3
#define PIN_S2 4
#define PIN_S3 5
#define PIN_COLORSENSOR_OUT 6
#define PIN_CON_WALLABY 9

#define PIN_LIGHTSENSOR A0

Servo servoMotor;
```

Figure 7: Code of Pindefinitions and Libraryimplementation

This object is now linked to the pin that was assigned earlier using servomotor.attach(PIN_CON_WALLABY);

```
void setup()
{
pinMode(PIN_S0, OUTPUT);
pinMode(PIN_S1, OUTPUT);
pinMode(PIN_S2, OUTPUT);
pinMode(PIN_S3, OUTPUT);
pinMode(PIN_COLORSENSOR_OUT, INPUT);
pinMode(PIN_CON_WALLABY, OUTPUT);
servoMotor.attach(PIN_CON_WALLABY);
// Setting frequency-scaling to 2%
digitalWrite(PIN_S0, LOW);
digitalWrite(PIN_S1, HIGH);

pinMode(PIN_LIGHTSENSOR, INPUT);

Serial.begin(9600);
}
```

Figure 8: Code of setup

After that the servo could be controlled using servoMotor.write(value). The value is possible in the range from 0 to 180 and its unit is degrees. This function was used to turn the Servo in between a 90-degree angle in order to sort the poms. When the Servo is at 70 degree the red poms fall through the right whole. When the Servo is at 160 degree the green poms fall through the left whole. Because of this system, the differently colored poms can be sorted.

The code for the pom separation:
At first the program reads out the values for the color frequencies twice in order to minimize wrongly read values. After that the average is calculated.

```
color();
int redFreq1 = redFreq;
int greenFreq1 = greenFreq;
color();
int redFreq2 = redFreq;
int greenFreq2 = greenFreq;

redFreq = (redFreq1 + redFreq2) / 2;
greenFreq = (greenFreq1 + greenFreq2) / 2;
```

Figure 9: Code of refinement of color determination

Then the program determines whether the values are too similar – when the quotient of the red and green frequency is in a range between 0.8 and 1.2 – and does not move the servo. If the red frequency is higher than the green, the servo turns to a 70-degree angle. If the green frequency is higher the servo turns to a 160-degree angle. The delays are needed for the time between the pom getting recognized and the pom before falling down the transporter and reaching the servo where the poms are separated.

```
if (redFreq / greenFreq > 0.8 && redFreq / greenFreq
< 1.2)
{
        Serial.print("too similar");
}

else if (redFreq > greenFreq)
{
            delay(250);
            Serial.print("red recognized");
            digitalWrite(PIN_CON_WALLABY,
            HIGH);
            servoMotor.write(70);
}
else if (greenFreq > redFreq)
{
            delay(250);
            Serial.print("green recognized");
            digitalWrite(PIN_CON_WALLABY,
            LOW);
            servoMotor.write(160);
}
```

Figure 10: Code of color determination

IV. RESULTS

Several problems concerning the sorting system and color recognition were identified. From the beginning it was obvious that a sensor was needed, someone without much experience in this area might not know that an IR light sensor is not an ideal choice for a color sensor. Although they are not recognizing the difference between green and red, they do work well for black and white. For that specific reason, the XHT 3085/3 uses it in the line following system. Some experimentations concluded, that a TCS3200 would be a much better choice for color detection.

As a matter of fact, the TCS3200 also was not perfect. The first reason was identifying the colors fast enough. A very difficult situation that took quite some time to solve was to recognize the colors correctly. One of the reasons it took so long was a wrong value in the informational table. In order to solve it multiple values needed to be tried out to find the correct one.

After finally solving that, options to connect the sensor to the Wallaby had to be thought through, since it did not have a proprietary connector. The sorting system that was planned did not have enough space for the new sensor. Therefore, the entire system had to be redesigned to include the replacement.

Lastly the servo needed to be connected, which turned out more difficult than expected, because of the missing knowledge that a servo needs an additional power source. By branching off the ground cable and the 5V cable and connecting them to the Wallaby the problem was solved and the sorting could begin.

V. ACKNOWLEDGMENT

This outcome would not have been possible without our teacher and organizer Johannes Tomitsch. He made the participation on the ECER possible and helped us with some advice for the color sensor TCS3200. Special regards to the HTL St. Pölten for letting us use the laboratory room.

VI. LITERATURVERZEICHNIS

[1] C. f. Recycling. [Online]. Available: https://recyclingpartnership.org/communitiesforrecycling/recycling-how-it-works/. [Zugriff am 02 04 2022].

[2] KIPR, „Botball IR (TopHat) Sensor,“ [Online]. Available: https://botball-swag.myshopify.com/collections/sensors/products/lg-ir-top-hat-sensor. [Zugriff am 02 04 2022].

[3] M. Integrated, „Definition for Light Sensor,“ [Online]. Available: https://www.maximintegrated.com/en/glossary/definitions.mvp/term/Light%20Sensor/gpk/1221#:~:text=Light%20sensors%20are%20a%20type,or%20convert%20light%20to%20electricity. [Zugriff am 02 04 2022].

[4] Keyence, „What is a Color sensor?,“ [Online]. Available: https://www.keyence.com/ss/products/sensor/sensorbasics/color/info/. [Zugriff am 02 04 2022].

[5] WatElectronics, „What is Color Sensor,“ [Online]. Available: https://www.watelectronics.com/what-is-color-sensor-working-its-applications/. [Zugriff am 02 04 2022].

[6] E. 360, „Color Sensors Information,“ [Online]. Available: https://www.globalspec.com/learnmore/sensors_transducers_detectors/vision_sensing/color_sensors. [Zugriff am 02 04 2022].

[7] Distrelec, „TCS34725,“ [Online]. Available: https://www.distrelec.at/de/rgb-farbsensor-tcs34725-5v-adafruit-1334/p/30091138. [Zugriff am 02 04 2022].

[8] AZ-delivery, „TCS3200,“ [Online]. Available: https://cdn.shopify.com/s/files/1/1509/1638/files/Farbsensor_Datenblatt.pdf?13859665781781208664. [Zugriff am 02 04 2022].