IOGS ProTIS 2021. Technical report. System for color detection and sorting of objects on a conveyor belt.

Igor RESHETNIKOV, Ruiyang HUANG, Théo MARTIN,
Raphaël DE CHEVRON VILLETTE, Mike RAYNAUD

11th May 2021

Contents

1 Overview of the system

As indicated in the fig.1, the system could be divided into three parts:

1) At the bottom the *mechanics*, including:

a) a conveyor belt, driven by a stepper motor, to bring the color blocks;

b) a rotating hump, driven by a precise servo-motor, to sort the blocks;

c) boxes to store the sorted blocks;

2) In the middle the *Raspberry Pi*, which receives images from the camera, processes them, and sends commands to the sorting hump and the light source;

3) Above the imaging system, including:

a) a LED light source to illuminate the blocks in a quasi darkroom to promote the stability of image quality;

b) a web-cam to image the blocks during their passage through the darkroom in real-time.

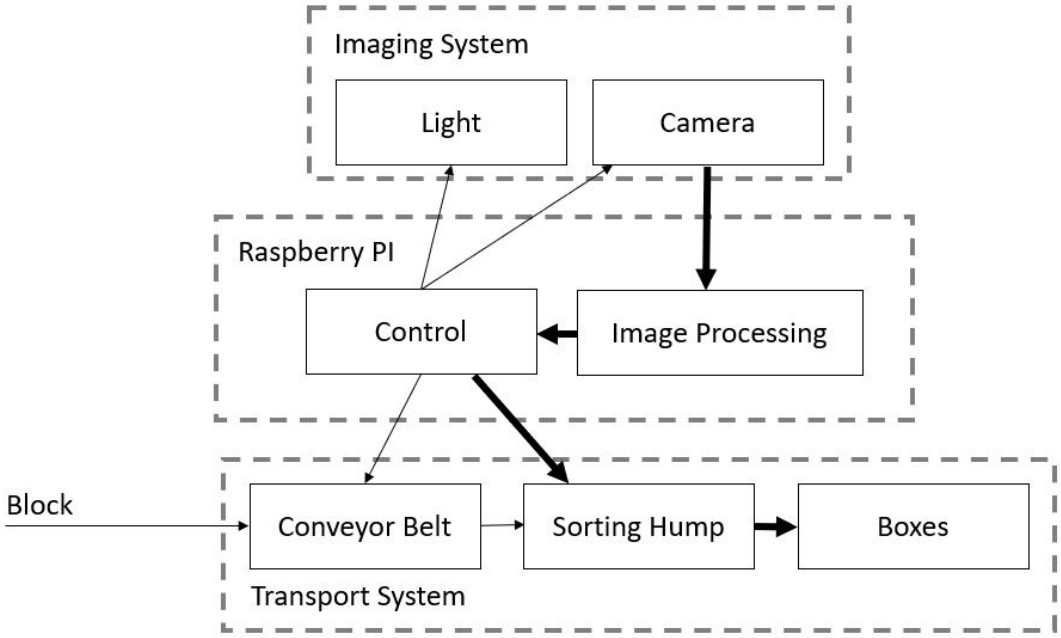

Figure 1: Bloc Diagram

2 Hardware specifics

2.1 Power supply

Most parts of the system should be supplied by a constant 5V source, except the servo-motor which is commanded and supplied totally by the Raspberry Pi. The stepper motor and the LED light source consume most of the energy; the others are digital circuits.

2.2 Conveyor belt

The conveyor belt used in this project is a Dobot conveyor belt from Dobot Conveyor Belt Kit (https://www.dobot.cc/products/conveyor-belt-kit-overview.html). It is driven by a stepper motor. With a combination of chip L297 and chip L298 driven by a clock, the DC supply is transformed to alternating square wave to drive the stepper motor.

The clock frequency determines the speed of the stepper motor, up to 600Hz to obtain maximum speed of the belt. It's recommended to use a signal generator to give the clock, but it can be as well driven by the Raspberry Pi using a PWM output port.

The switch of chip L297 could be commanded by a digital signal; it is connected to Raspberry Pi and could be switched on and off on software interface.

2.3 Darkroom Imaging

The blocks, put on the belt, would be brought to a small *darkroom* set at the end of the belt. A LED light source is suspended on the wall of the room to illuminate the blocks inside. (It's even better to have two light sources in order to get rid of the shadow.) The LED is connected to a MOSFET, through which the Raspberry could control the source. The power LED consumes at maximum brightness is around 8W. One could reduce the power by adding a resistance in the supplying circuit.

The darkroom also has a small window for the camera to picture the blocks in real-time, the images are sent to Raspberry Pi so that the color is identified. There exists a delay of 2 seconds or so between the real-time image and the reaction from Raspberry Pi; however, the delay helps so that the hump reacts only when the blocks fall onto it.

2.4 Block Sorting

The webcamera filming the darkroom works all the time, as well as the conveyor belt. Not the full FOV of the camera is used in the color detection. Only a small rectangle in the center of the camera's FOV is used to detect colors.

When there are no blocks in the detection rectangle, the camera detects black color (it is the purpose of the darkroom), so no command is sent to the sorting hump. When the block at the conveyor belt comes to centre of the darkroom, the camera detects the presence of the block and identify it's color from the pre-installed list of colors. In current configuration, the system can identify 4 colors: red, blue, green and yellow. The simplicity of the written code allows to add more colors to the list with adding only a couple lines of code. The color identification process is described in the corresponding section.

After the color is detected, the Raspberry Pi sends the command to the sorting hump to turn to the corresponding color box. The hump is based on the servomotor with full angle of about 180 degrees. So, each color corresponds to the certain angle of the servomotor. The matching between colors and angles of the servomotor are defined in the program.

2.5 Parameters

Stepper Motor	
V_{L297}	5V
V_{Stepper}	5V
I_{Stepper}	$\sim$0.5A
Clock Frequency	100$\sim$600Hz
Servo Motor	
V_{Servo}	5V
PWM Frequency	50Hz
PWM Signal width	0.5$\sim$2.2ms
Rotation Time	$<$300ms
LED source	
V_{MOSFET}	5V
V_{LED}	5V
P_{LED}	$<$8W

2.6 Circuit diagram

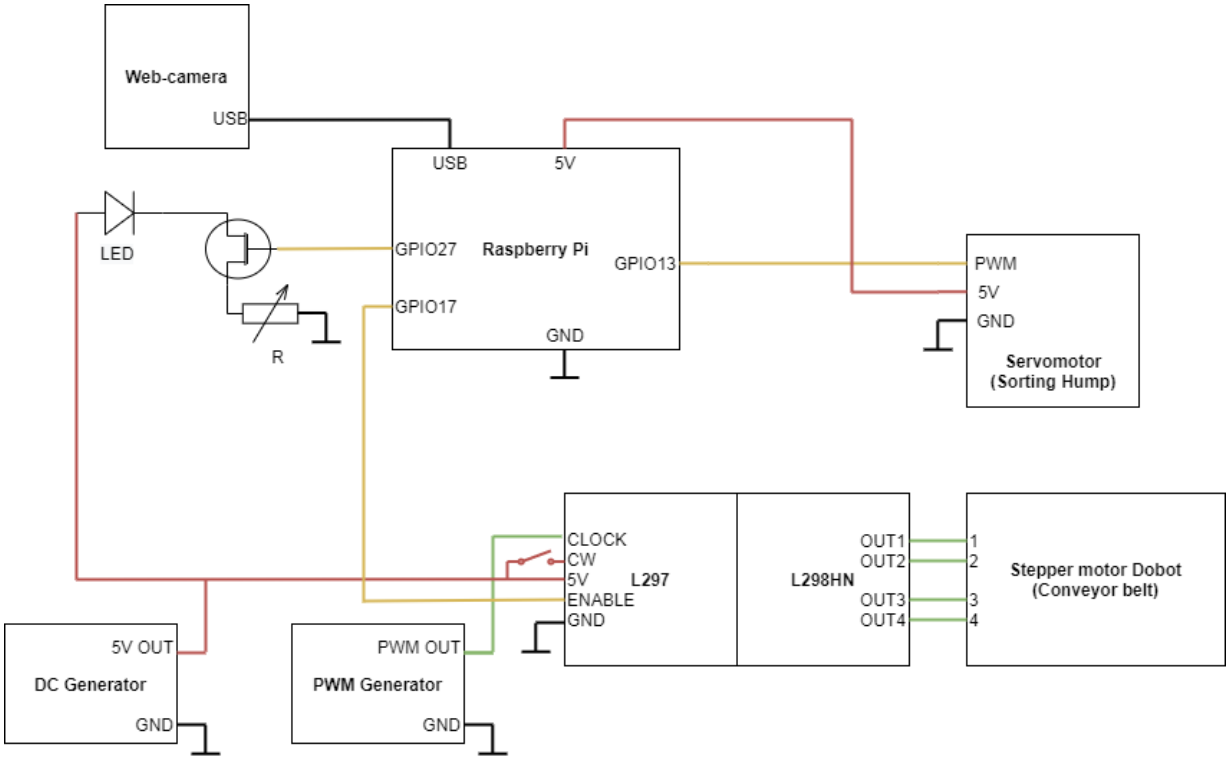

Figure 2: Circuit diagram.

Web-camera is connected to the Raspberry Pi via USB. Servomotor of the sorting hump requires ground connection, 5V DC signal and control PWM signal from GPIO13 from Raspberry Pi. LED for the darkroom can be controlled via Raspberry Pi and MOSFET. It requires 5V DC and control signal from GPIO27. Adjustable resistance R can be changed to set the brightness of the LED.

Stepper motor Dobot connected to the boards L297 and L298HN. The DC 5V is supplied by external DC generator. PWM signal, that control speed of the conveyor belt, is supplied by external PWM generator. Conveyor belt can be enabled and disabled manually via Raspberry Pi GPIO17 connected to the ENABLE port of the L297. Optionally, CW port of the L297 can be connected to DC 5V in order to change the direction of the belt.

3 Image detection

All the software part was done in Python, and in particular using the Tkinter library for the interface and the OpenCV library for image processing.

In order to optimally detect the colors of the cubes, we decided to study the images in HSV (Hue, Saturation, Value) and not in RGB. Only the hue plays on the color itself, the saturation and the value modify respectively the intensity and the brightness. Hue is a number that varies like an angle and which curls. Thus, in order not to have any problem with colors close to 0 ° = 360 °, which corresponds to red, the opposite color (180 °, ie cyan) is detected on an image which has been previously inverted.

A calibration phase is necessary in order to calibrate the colors that will be recognized according to the configuration and the surrounding light. These calibrated values can be saved in a document so that they can be reused even after closing the interface. They can thus be loaded at any time.

The color detection is therefore done at the shade plus or minus a precision value which can be modified directly in the interface. A mask verifying the preceding condition is then produced making it possible to see only the detected pixels.

In order to detect an entire cube of one color, an function from OpenCV named 'erode' is used to remove any stray pixels that might have been detected due to color noise. This makes it possible to have only one related block of color detected. A second OpenCV function called 'dilate' then allows you to expand the group of pixels that have not been removed in order to find the original area. These two functions take iteration values as parameters which can be modified in the interface. The image is also slightly blurred in order to stabilize the detection.

During a detection, the time of detection and the color detected are filled into a log file. The interface also enables to start and stop both the conveyor and the light at any time