G1-Detailed Design - Stackbot
03/06/2013

Group Members:
Qi Zhiyuan
Sanjay Mario Dmello
Jovita Castelino
Anitha Ganesha

System Design Overview:

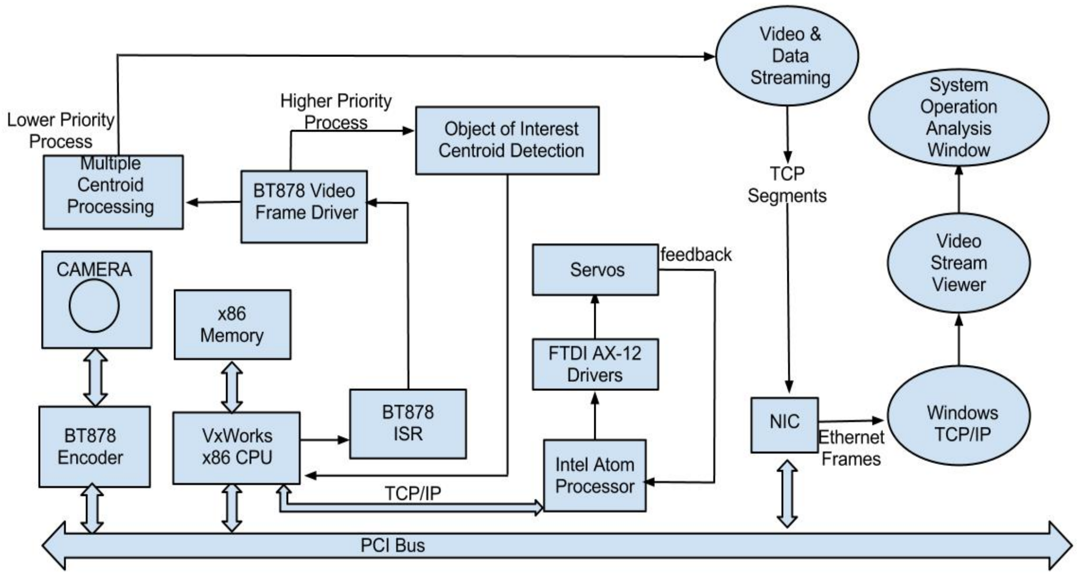


Fig 1: Detailed System Design

1. Hardware Components
1.1) Conveyor Belt System
The conveyor belt system is designed to be tunable by changing the driving voltage of the DC motor. The image processing system in combination with ATOM can detect the moving speed of the objects on the belt and send the velocity information to an 8051 microcontroller. The microcontroller captures the signal and calculates the required duty cycle for the PWM signal. The PWM signal then drives a TC 4428 gate driver and feeds an amplified signal to the MOSFET of the Buck converter. The Buck converter gives out a suitable voltage for the motor that drives the conveyor belt. In order to achieve high level DC gain and low level damping in the feedback control, we may use a PID control circuit for the Buck converter.

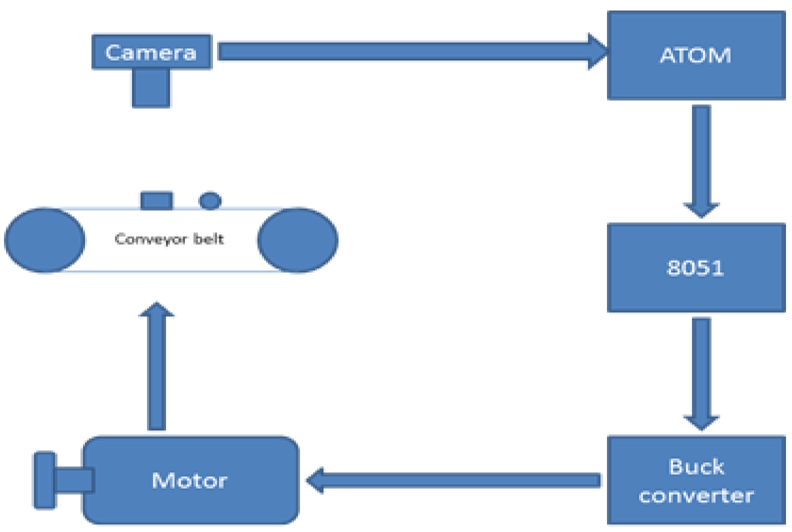


Fig 2: Conveyor Belt and its driving system

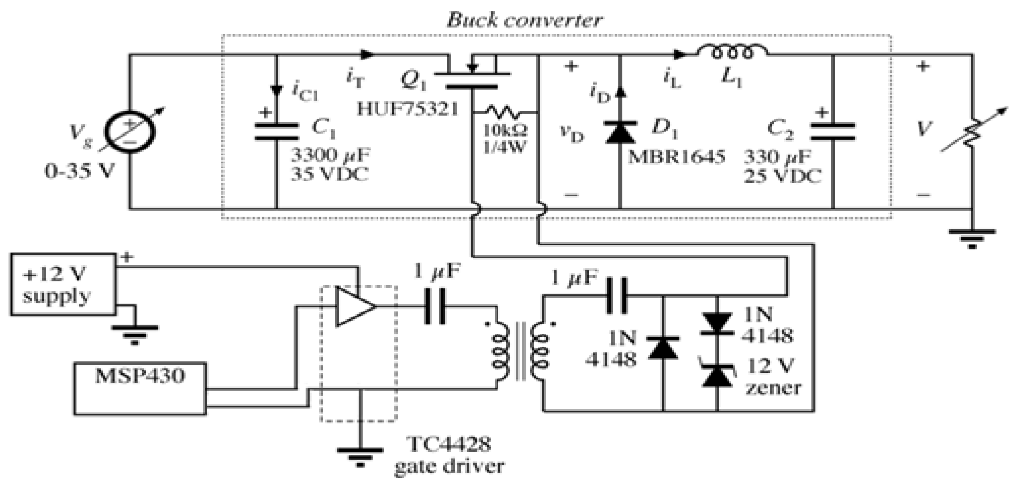


Fig 3: Buck converter for tuning the speed of the conveyor belt

2. Software elements:

2.1) Task 1: Capturing Frames

A frame grabber will be implemented which receives the captured images of the objects moving on the conveyor belt. These frames are stored in a buffer which can be later accessed by the video frame driver to process the video frames. Analogous to the btvid_driver function in the code; the functions we will be implementing are as follows:

- start_video_report(): returns a pointer to the buffer when a new frame is received.
- start_video(): This is the entry point function for the btvid_driver function. Initialization of the btvid_driver configuration, PCI device configuration and testing takes place here.
- set_mux(): To set the input obtained from the NTSC video camera
- interrupt_handler(): Implemented in the Frame Event ISR.

- frame_rdy_semgive(): provides frame services for the Frame Sequencer.

2.2) Task 2: Image Processing

This task is responsible for the image processing.This is responsible for detecting the centroid of the object of interest after detecting the shape of the object.

- readFrame() : This function is responsible to read the frame contents after it is processed by the video frame driver and and store the contents in a local buffer.

- filterFrameContents(): This function selectively filters a part of the frame which contains the object of interest and provides this as the input to the following functions

- centroidComputation(): This function is used to find the centroid of the objects within a frame of image. Firstly, we set-up a fixed intensity threshold and a fixed size threshold. Then only pixels with higher intensity than the threshold will be considered. Also, the size threshold will be applied to select the object of interest. The centroid of objects is calculated via the following equations.

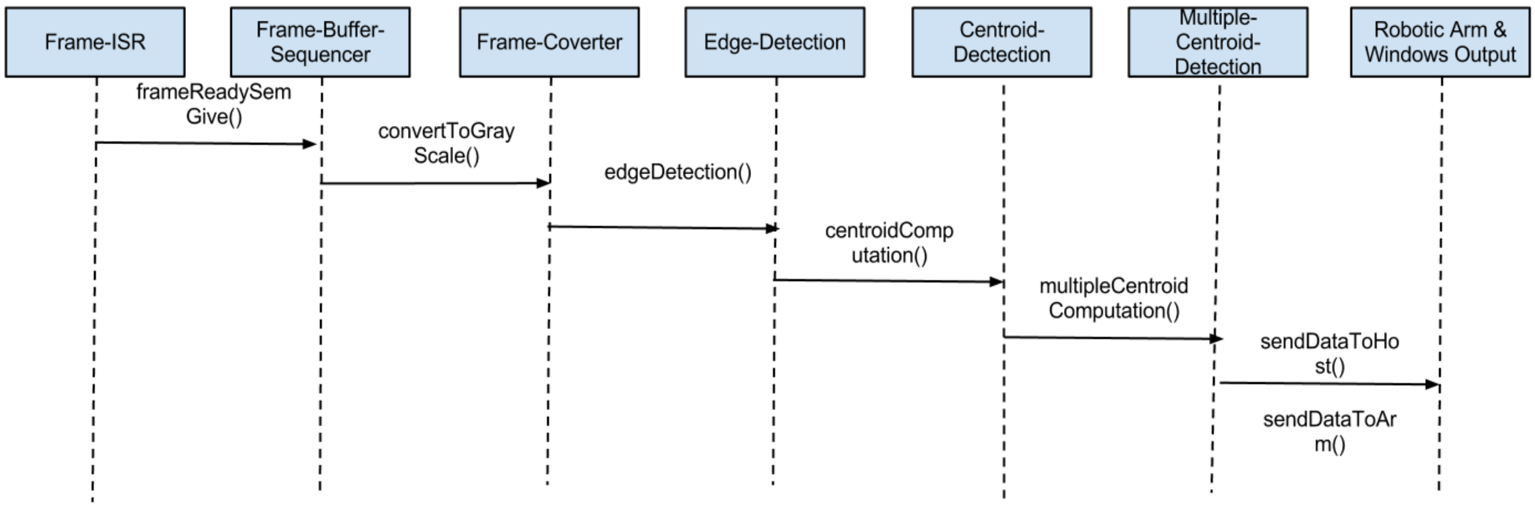
- edgeDetection(): This function can detect and store the edge information of the objects within a frame. Canny's method of edge detection could be applied. Generally speaking, there are two steps for edge detection.

  Firstly, the raw intensity data is calculated with a smoothed Gaussian module for convolution. The data obtained is blurred in comparison with the original one. But the advantage is that the noise from a single pixel will have little effect on the intensity of the images.

  The second step is tracing the edge of the data obtained in step one. Two intensity gradient thresholds, low and high, will be applied for edge tracing purpose. The higher threshold will run first and gives out a detectable edge. Based on the output of the higher threshold the lower threshold can be utilized to find out the whole edges of the objects.

- velocityEstimation(): This function gives out the velocity of the moving object in real time. The centroid information from the four frames of images is used for the least square fitting of the velocity. The residual sum of squares is minimized for the purpose of computation of the average speed of the conveyor belt.

- multipleCentroidComputation(): This function is responsible for computing the centroid of multiple objects that are captured in a frame. It utilizes the same algorithm as discussed above in the function centroidComputation().



2.3) Task 3 : Windows Display:

- sendDataToHost(): This function is responsible for sending the data computed in the previous function to windows based system for display through TCP.

- displayCentroid(): This data which contains the information of the multiple centroid of different objects captured in a video frame once received from the vxWorks is displayed on the host.We also need to ensure that a new display is seen for every frame captured.

2.4) Task 4 : Robotic Arm control:

The Atom Processor drives the Robotic arm and the Atom board is loaded with code to perform the robotic arm movement functions.
Basic functions which are to be included in the Servo Control Code are as follows:

- init (): The AX-12 actuators communicate serially with the servo controller. The initial position of the arm is set using this function.

- calc_param(): This function involves calculation of actual data like the alpha, beta and gamma values, torque, direction et al.
  So , it would seek the required data from the VxWorks machine via TCP/IP and perform the calculations. These are different parameters to be fed into and read from the RAM / ROM of the AX12 actuators. This function will also contain information about the stack position and its size to calculate dropping height and co-ordinates.

- actuation (): This function would involve the serial transmission of calculated parameters into the actuators via the servo controller. The packets would be framed and status feedback will be obtained from the actuators.

- pick_status(): This function has a count of the number of attempted pickups on a single object.
  The conveyor belt is slowed down if the object is missed consecutively. A response from the VxWorks system is the value passed to this function to determine success or failure of the object pickup task.