Unified Robotics III - Manipulation
Team 11 - Final Report

Evan Bosia[1] and Lumbini Parnas[2]

***Abstract*— This paper discusses a control system design used on an automated 2-Degree of Freedom (DOF) arm to pickup blocks moving on a conveyor. The design relies on a Proportional Integral Derivative (PID) controller along with forward and inverse kinematics to move the end effector to the required position. Block sensing is done by using two infrared sensors and the weight is determined using current sensing.**

I. INTRODUCTION

This project was completed in the Unified Robotics III course (RBE 3001) at Worcester Polytechnic Institute (WPI). The overall course focus of RBE 3001 is the control of different types of robotic arms. The goal of this project is to develop a control system for a 2-DOF robotic arm using the AVR644p Microprocessor. The 2-DOF arm is used in a simulated assembly line, where the system is required to detect and pick up blocks from a conveyor belt. In addition, the system is able to determine the weight of the blocks and sort accordingly. Low level embedded programming was the primary method in which the AVR644p Microprocessor and the included peripherals were configured.

In order to achieve the goal efficiently, we developed the following approach plan. The first step was to implement a communication system between the AVR microprocessor, the peripherals-to acquire sensor data and MATLAB for any calculations. This allowed us to implement the software for the physical system using forward and inverse kinematics. The Proportional Integral Derivation (PID) control system was used to precisely manipulate the arm. Finally, 2 Sharp GP2D12 Infrared Range Finders were used to detect the location of the block on a motorized conveyor belt moving at a constant speed. This paper will outline the kinematics, signal processing, overall program structure used in successfully accomplishing the task at hand.

The sensor information was then processed to calculate the moment at which the arm should be moved to pick up the block. This entire software was designed to act as a state machine allowing us to control the states individually and have the processes be independent of each other; this allowed for smooth testing of the program.

II. DESIGN

A. Arm Control

A PID controller is used to ensure the arm is positioned at the required joint angles. Our PID function takes in an error term and converts it to a usable output, which in our case was motor torque. Equation 1 is the infinite case, and Equation 2 is the discrete case. We use the discrete case in our program.

$$u = K_p * e(t) + K_i * \int_0^t e(T)dT + K_d * \frac{d}{dt}e(t) \tag{1}$$

$$u = K_p * e(t) + K_i * \sum_0^t e(t) + K_d * (e(t) - e(t-1)) \tag{2}$$

The PID constants K_p, K_i, and K_d are all changed within the program, allowing for the control of the arm to be tuned for different needs. For example, when moving to the home position, the value of K_i is raised significantly. Here it is more important for the arm to settle at the correct position rather than having a quick settling time in order to ensure absolute measurements. By increasing K_i, the settling error can be greatly reduced.

The controller uses encoders to generate the error term. Encoders are used for PID instead of potentiometers because encoders are digital; there is no noise on encoders when stationary. Noise has a negative affect on PID, especially with the derivative value. Any jumps in error values caused by noise will cause the derivative value of the PID equation to be incorrect. Combined with the the sampling rate of 100Hz and any high K_d values, this noise can be amplified enough to have negative effect on the motor signals.

Because encoder values are relative to where they were last reset, the *homePos()* function is used at the start of the program to ”zero” the arm. To calculate how to move the arm to its zero position the function first reads a series of potentiometer values at each joint, taking the median to get absolute joint angles. Using these joint angles as a reference, the robot can then use encoder driven PID control to move the arm to its zero position. Once the arm reaches this position the encoders are reset, allowing for absolute angle measurement using the encoders.

B. Block Sensing

Two IR sensors are used to determine the location of the block on the conveyor. Each of these sensors are located perpendicular to the movement of the conveyor belt. Despite the similarity of placement, each IR sensor is used for a different application.

The first IR sensor along the conveyor is used to determine the position of the block laterally on the conveyor belt. The direct readings from the IR sensor are measured using the ADC, meaning its value is output in ADC counts. In order to get an accurate positional measurement from the IR sensor, the sensor was linearized over the expected range of the block.

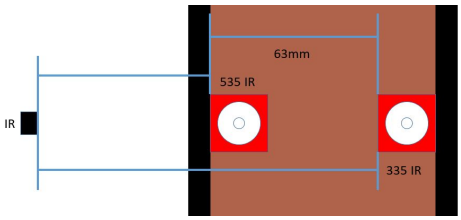

Fig. 1. Method of linearizing the IR sensor

By measuring the close IR value of the block, the far IR value of the block, and the difference between these two points in millimeters, a millimeter per IR value constant could be created. This relationship was eventually use to determine absolute position equation in relation to the robot base.

$$260 - ((Measured - Far) * \frac{Distance}{Far - Close}) \tag{3}$$

Plugging in values...

$$260 - ((Measured - 335) * 0.315) \tag{4}$$

Equations 3 and 4 assume that the infrared sensors measure distance linearly. While this is false, over the short interval the non-linearity has little effect on the overall accuracy of the readings.

Once the block is detected, multiple readings of the IR sensor are used to determine the position of the block. The program takes multiple readings to eliminate noise from the analog read. The median of these readings is used to calculate lateral position of the block.

The second IR sensor is used as a rudimentary trip sensor. Once the block is detected by this IR sensor, the arm prepares to pick up the block. Because the conveyor runs at a constant speed, the robot can use the trip sensor as an indication to switch to block pickup mode and use delays to pick up the block.

C. Forward Kinematics

We used the coordinate frame setup in Figure 2 for forward and inverse kinematic calculations on the robot. θ_1 and $e\theta_2$ are also used in our PID calculations. Note that θ_1 and θ_2 are positive in the counterclockwise direction.

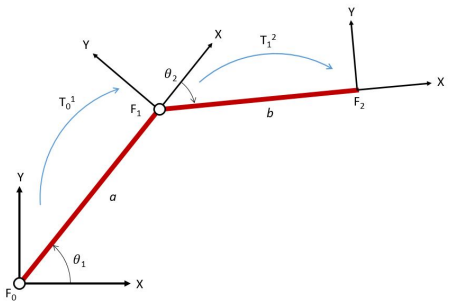

Fig. 2. Coordinate frames and transformations

The forward position kinematics are as follows:

$$X = a * \cos(\theta_1) + b * \cos(\theta_1 + \theta_2) \tag{5}$$

$$Y = a * \sin(\theta_1) + b * \sin(\theta_1 + \theta_2) \tag{6}$$

These equations are not used directly in the AVR program. Instead, we used them initially to check the expected positions of the arm when testing arm control.

D. Inverse Kinematics

Inverse kinematics are used to generate the correct joint angles for the arm to move the gripper to a Cartesian position with respect to the base. The following inverse kinematic equations are used to solve for these angles:

$$\cos(\theta_2) = \frac{X^2 + Y^2 - a^2 - b^2}{2 * a * b} \tag{7}$$

$$\theta_2 = \pm atan2(\sqrt{1 - \cos(\theta_2)^2}, \cos(\theta_2)) \tag{8}$$

$$k_1 = a + b * \cos(\theta_2) \tag{9}$$

$$k_2 = b * \sin(\theta_2) \tag{10}$$

$$r = \sqrt{k_1^2 + k_2^2} \tag{11}$$

$$\gamma = atan2(k_2, k_1) \tag{12}$$

$$\theta_1 = atan2(Y, X) - atan2(k_2, k_1) \tag{13}$$

In order to determine which position the arm is in, we assumed that arm is elbow down until θ_1 is less than 0. In that case, we used an if statement to change the angles so that the arm would be in elbow up position. γ was solved for in order to convert the angle. All of these equations are run separately in MATLAB, with serial communication connecting the AVR and MATLAB which is detailed in the II-G section.

E. Weight Sensing

The LMP8601 current sense circuit is used to determine the weight of the block. The current of the upper motor is measured. We chose the upper motor as the measurement source, because even though using the lower motor in theory would cause a greater torque change on result of the larger radius, using the upper link worked more effectively.

PID control is disabled for the duration of the current sense measurement. The varying signals of PID control cause varying current sense readings which effectively negates the effectiveness of current sensing, as it acts similar to noise. The upper link motor is instead set to a constant voltage so that the current reading is only affected by the torque of the link.

Current sensing itself is very noisy. In order to get an accurate reading, 50 samples are taken from the current sense circuit and the median is used to filter out extraneous values.

These readings are compared against a threshold to determine what type of block is being measured. Even with these steps, the current sense value is not accurate enough to determine the weight of the block. For this application where only determining between light and heavy was important, using current sensing works just well enough to be consistent and effective,

F. Software Structure

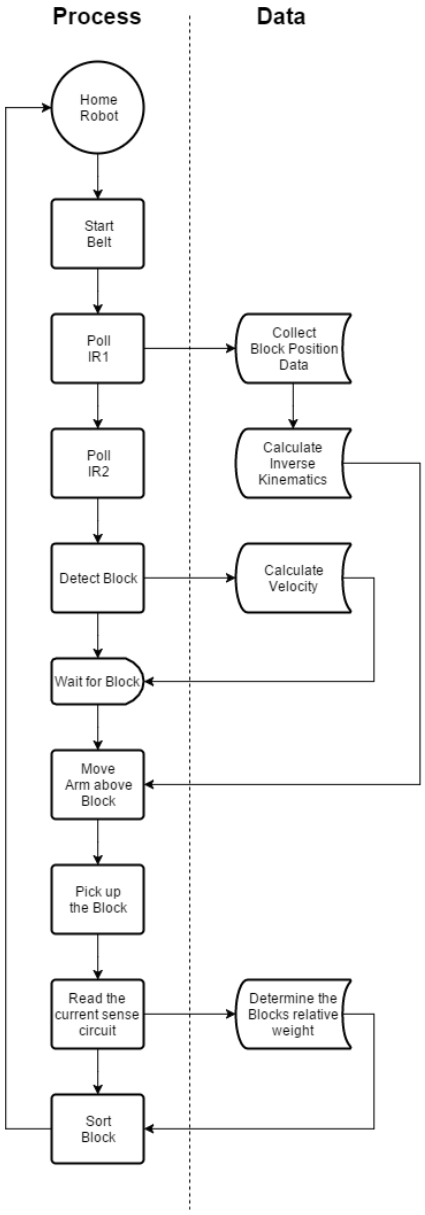

Fig. 3. Program Structure Flowchart

The overall structure of the program is a state machine. Enumerated states were used to represent each process represented in Figure 3. As the program completes a process, the state changes and the robot works on the next process. All of this logic is held in *stateFunction()* which is constantly being called by a loop in *main()*.

G. MATLAB Integration

The main link between MATLAB and the AVR is done through bitwise communication. Because there are multiple types of data that need to be sent to MATLAB (joint angles for real time plotting, distance of block for inverse kinematics), a character is sent prior to the data transmission. MATLAB is able to see this character and understand what data is being passed in. MATLAB also updates the states being displayed on the real time plotting depending on the character received from AVR. There is also a character that requests joint angles from the inverse kinematics function so that the robot knows where to move. This data iss sent within states in the *stateFunction()*.

H. Overall Setup

Making sure the robot is consistently setup is very important for this application. If the arm shifted even by a degree, picking up the blocks could fail. We designed and created a laser-cut base board to correctly align the arm, the IR sensors, and the conveyor. A top down view of the base board is shown in Figure 4. The base board with the IR sensor mount and arm is shown in Figure 5

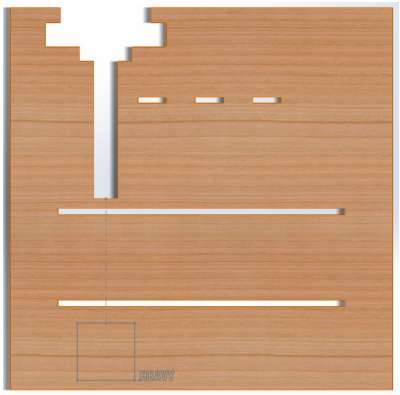

Fig. 4. Base plate top view

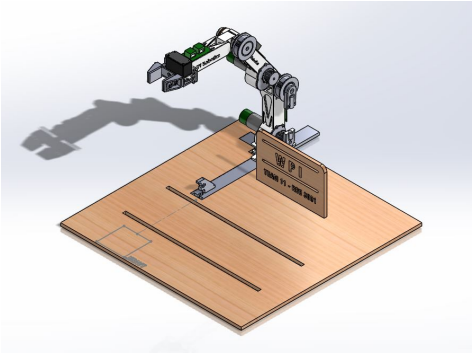

Fig. 5. Base plate with IR sensor mount and arm

III. Results Discussion

A. Block Sensing

Block sensing worked for the most part, but there were instances where the IR sensors needed to be re-calibrated.

These problems occurred when setting up the arm. One possible cause to this problem could have been the angle of the IR sensors. The sensors are only attached to a thin piece of wood with little support along the IR sensing direction. While the piece was eventually glued down, there still could have been slight changes in angle of the sensors, which would change distance readings. We had some problems with this, but once we glued the IR sensor mount down we were able to get it to produce more consistent results.

B. Block Pickup

The success of Block Pickup highly depended on the success of the *homePos()* function. If the *homePos()* function ran correctly, as in the arm was zeroed when the encoders were reset, the PID control of the arm worked well enough to move the arm to the correct positions. Without being properly zeroed, the arm would instead miss picking up the blocks. Because encoders give relative positional measurements, if the *homePos()* resets the encoders not at the actual zero position, that angle error is carried throughout the rest of the arm operation. We had about 75% success rate using the *homePos()* function, which could be improved in the future.

C. Weight Sensing

Weight sensing worked very consistently. By manually setting the motor value to a constant and taking the median of 50 current sense readings, the robot is able to determine the weight of the block and sort accordingly. In order to make current sensing precise, a future implementation can include a filter to de-noise the signals received from the current sensor before used for processing. Other weight sensing methods like using a load cell on the end effector would enable for either comparative measurements or at least less noisy results.

IV. CONCLUSION

This project resulted in a successful implementation of the RBE3001 Board along with the 2-DOF arm to complete the simulated assembly line task. It helped us develop skills in embedded programming as well as implementing basic PID and kinematic control on a 2-DOF arm. Most importantly, the testing procedure taught us to debug and troubleshoot the system. The various techniques used in this course are relevant and expandable to industrial applications which use robotic arms.

ACKNOWLEDGMENT

We would like to extend our gratitude to Professor Fischer, Nolan Poulin, Nathaniel Goldfarb, the SA's and especially Joe St. Germain - We love you.

REFERENCES

[1] J. St.Germain, "Rbe 3001 board rev2," 2013, robotic Engineering, Worcester Polytechnic Institute.
[2] Atmel, "8-bit atmel microcontroller with 64k bytes in-system programmable flash," 2013, aTmega 644/V.
[3] J. S. Eric Willcox III, "Rbe 3001 development board guide," 2013, robotic Engineering, Worcester Polytechnic Institute.
[4] S. C. T. Staff, "Gp2d12 optoelectronic device data sheet," 2005, sHARP ELECTRONIC COMPONENTS and DEVICES.
[5] A. Robotics, "Linearizing sharp ranger data," 2007, linearizing Sharp Ranger Data.
[6] O. Dr. Rainer Hessmer, "Kinematics for lynxmotion robot arm," 2009.

LAB 4 – FINAL PROJECT:
AUTOMATED PRODUCTION LINE

TEAM MEMBERS

LUMBINI PARNAS EVAN BOSIA ANDRE IMPERIALI

DESIGN SPECIFICATION GRADE (TEAM EFFORT)

_____ _____ _____

IN-LAB SIGNOFFS

TASK	*INSTRUCTOR/TA*	*DATE/TIME*
Demonstrate calibrated IR sensor (20 pts)	NIRP	10/8/16
Real-time visualization of arm in Matlab (20 points)	NIRP	10/8/16
Control of arm through Matlab (20 points)	NIRP	10/8/16
Demonstrate inverse kinematics and control of servos to pick up block (20 pts)	NIRP	10/8/16
Demonstrate block sorting by weight (20 pts)	NIRP	10/8/16
Final Project Demonstration (30 pts)	[signature]	10/13/16

** Attach this sheet to your report.*

ADDITIONAL REMARKS