Blackstone Valley Regional Vocational Technical High School
Student Portfolio- Project Reflection

Date	**3-1-23**
Student Name	**Om Patel**
Subject	**CIM (10th)**
Instructor's Name	**Mr. Oliveira**
Instructor's Signature	**Mr. Oliveira**

In the first part of this project, our main task was to build a transfer system using only VEX parts and RobotC. However, we didn't just have to make a transfer system that transported one part from one side to another. This conveyor had to follow many other constraints and requirements. Some of these constraints include, that there must be an emergency stop button that will stop all operations no matter what. A few more requirements include, the system must be at least 12 inches long, must start with a digital input, and must make two stops before being dumped into the bin at the end of the system. These two stops had to be automated and allowed for a block to be removed at them and once the block is returned to the system the system will automatically start again. The final requirement for the system was for it to be able to start again when the start button is pressed again, allowing the system to run again.

After learning the outlines for this project, my partner and I when straight to brainstorming ideas on how to make an efficient and precise transfer system. We thought of many ideas but landed on using two different types of sensors an encoder and two ultrasonic range finders. The encoder accounts for the number of degrees a shaft moves while a range finder finds the range between an object and itself. The two range finders would be located at the two stops at which the item will be picked up and placed back. One issue with these sensors is that they aren't very precise and due to this, the encoder was picked to help. The encoder would allow for the block to be picked up at a certain range using the number or degrees to account for how far the track has moved on the conveyor belt. These two sensors paired allow for an accurate stop location and a way for the system to start and stop when a block is placed back and picked up.

After our brainstorming, my partner and I when to the building process in which we tried to make our ideas to life in the cheapest manner. For this project, each VEX part has a certain price and this too makes our team have to think about whether or not to use a part or to find a cheaper alternative. After making the bones of the projects, the metal, gears, and tracks, I worked on sketching the system we had created. While I did this my partner added all the sensors we had decided to use from our brainstorming session. With the transfer system being built and sketched the last item on our list was to code it and create the flowchart.

The first thing my partner and I did even before making out code was to make a flowchart. This would allow us to get an idea of how and what we need to code. The code was somewhat difficult as my partner and I tried more complex techniques that didn't work. After lots of trial and error, we remembered the "waitUntil" command. This command allows for the code to stop until a certain task has been performed. In our case, it was for the sensor to detect the block and be in a certain range of the encoder. This would then allow for the motor to stop and wait for the block to be picked up and placed back down and continue to move the conveyor. This pattern of commands would be used again for the second sensor. Then we would use the encoder again to allow the system to stop after the block has fallen off the conveyor belt. The final part of the code was our emergency stop and start button which would continue our code after it finished. The emergency stop was created by creating a separate task that stopped all tasks and then imported into the main block of code, allowing it to stop no matter where the code is in its sequence. For the start button, we used an "if" statement in which once the button is pressed the program would run, reset the encoder, and go through its list of commands.

In the second part of the transfer system project my partner and I have to code and teach 2 Lynx Motion Robots to pick up a block, place it on a conveyor belt, and do other various tasks. The first robot first has to pick up a block and place it onto the start of the conveyor belt we made in the first part of the project. The robot will then send a signal to the VEX Cortex which will start the motor until the sensor senses the block at its first stop. The Cortex will then send a signal back to the first robot and in which it starts its new task. This task consisted of picking up the block, placing it in a safe location, going home, going back to pick up the block, and placing it back to where it was on the conveyor belt. The robot would then send a signal back to the Cortex which runs the program again until the block reaches the second stop. The Cortex will then send a signal to the second Lynx Motion Robot who will then pick up the block, place it in a safe location, go home, go back to pick up the block, and place it back where it was on the conveyor belt. After the robot goes back to a safe position it will send a signal into the VEX Cortex and allow the motor to run again until the encoder reaches a certain value. When it reaches this value the block will be right at the end of the transfer system and send a signal back to the second robot. The second robot will then pick it up and place it back in the original starting position of the block. After all the tasks are done, the program can then repeat and loop till we want it to stop. The code also has to stop just like it did in the first part, with an emergency stop.

After learning the conditions of the second part I created a flowchart that will allow me and my partner to code and program both the conveyor and the Lynx Motion Robot. After the creation of the flowchart, I started to code the Cortex while my partner started programming the Robot, allowing for a time-efficient workflow. The Cortex’s code was a copy of the code from the first part but with a few differences. One difference was the Digital Input and Output settings for the ports. This would allow for the Cortex to detect inputs from other devices such as the Lynx Motion Robot and other VEX Cortexes. Instead of the Sonar waiting for the block to be placed back onto the conveyor belt, we can use the input command to allow for a command to run after the Robot sends an output. One quirk in this process was when using the Optocoupler to connect the robot to the Cortex it makes the signal a 0 rather than a 1 as it only sends low values. We also made the Cortex send outputs allowing the Lynx Motion Robot to do its tasks. The final change made to the code was we added a section where the block stops at the end of the conveyor belt rather than letting it fall off.

As I worked on my code my partner started working on programming the first robot this was simple but very tedious. Each placement has to be as precise as possible for it to run has how it was intended to. A lot of mistakes were also made such as forgetting to add a position above a position so it to not run into other objects or sending the output signal out of the correct port. But after these mistakes, my partner finished working on his robot right when I started to program the second one. There was one problem with my robot though, it was sporadic and had trouble not jittering, making it difficult to record positions. This was the worst part of the project but we were still able to make it threw it, even past technical difficulties with the FlowArm Program and the second robot. The final part of this project was to sketch the final contraptions and record a video of the robot working. With my robot being a bit troublesome, we had to take a couple of takes of the video, but after a few tries, we had done it and finished the project.

In conclusion, the transfer system project provided a valuable opportunity to learn and apply engineering principles to build an automated system using VEX parts, Lynx Motion Robots, FlowArm, and the RobotC programming language. Through this project, I learned how to work with different sensors like encoders and ultrasonic range finders and how to develop an efficient and precise system. I also learned how to think critically about the cost and functionality of each part while building and sketching the system. Through the coding aspect of the project, I used flowcharts to organize and create an image out of my thoughts making it easy to write the code. The second part of the project taught me how to integrate two Lynx Motion Robots into the Cortex. This interaction thought me how robots and systems communicate and how engineers have to think when creating complex systems. Overall, this project helped me to develop skills in problem-solving, critical thinking, collaboration, and patience which will be valuable in my future endeavors in the field of engineering.

Blackstone Valley Regional Vocational Technical High School
Student Portfolio- Project Reflection

Technical competencies and academic skills demonstrated by completing this assignment.

Framework Standard	Description
2.E.01.16	Design and create a program to evaluate data and make decisions using external digital and analog sensors.
2.E.01.17	Formulate a flow chart to correctly apply basic programming concepts.
2.E.01.18	Describe the function of sensors in electronic circuitry (temp., optical, etc.).
2.E.02.03	Program a robot or automated system to perform several tasks
Embedded Academics	**Description**
2.B.01.2.2	Demonstrate methods of representing solutions to a design problem, e.g.,, sketches, orthographic projections, multiview drawings.
2.C.09.7.3	Describe the advantages of using robotics in the automation of manufacturing processes (e.g.,, increased production, improved quality, safety).
2.B.01.02.8.d	Acknowledge new information expressed by others, and, when warranted, qualify or justify their own views in light of the evidence presented
2.C.20.9-10	Follow precisely a complex multistep procedure when carrying out experiments, taking measurements, or performing technical tasks, attending to special cases or exceptions defined in the text