



ASHESI UNIVERSITY

A Testbed for Teaching Industrial Automation 4.0

CAPSTONE

B.Sc. Computer Engineering

Delasie Koku Fui Fumey

2019

ASHESI UNIVERSITY

A Testbed for Teaching Industrial Automation 4.0

CAPSTONE PROJECT

Capstone Project submitted to the Department of Engineering, Ashesi University College in partial fulfilment of the requirements for the award of Bachelor of Science degree in Computer Engineering.

Delasie Koku Fui Fumey

2019

DECLARATION

I hereby declare that this capstone is the result of my own original work and that no part of it has been presented for another degree in this university or elsewhere.

Candidate's Signature:

.....

Candidate's Name:

.....

Date:

.....

I hereby declare that the preparation and presentation of this capstone were supervised in accordance with the guidelines on supervision of capstone laid down by Ashesi University College.

Supervisor's Signature:

.....

Supervisor's Name:

.....

Date:

.....

ACKNOWLEDGEMENT

I would like to thank God first and foremost for taking me through the years building up to this paper. I am most grateful for the aid received from my supervisor Dr Nathan Amankwah and also from Richard Akparibo. Finally, I would like to thank my family for the constant motivation they provided as this paper was being written.

ABSTRACT

This paper explores the current flaws of the undergraduate syllabi treating Industrial Automation and Control. It elaborates on the disparities between the current syllabi and what is used in Industry; particularly the current trend of Industrial Automation. It looks at the effect of not meeting this trend in Industry and thus offers a solution in the form of another curriculum, which incorporates concepts of Industrial Automation 4.0 which are necessary for the workplace. It uses the core topics of this syllabus to build a testbed to help familiarise students with how all these concepts work together. This testbed is simply a system to monitor and control the temperature and the flow of water between two containers using a programmable logic controller and an AC Drive. It concludes with the limitations of the curriculum and concepts which could be used to expand the curriculum further to facilitate other topics which are used in Industry.

Table of Contents

ACKNOWLEDGEMENT	iii
ABSTRACT	iv
TABLE OF FIGURES	0
CHAPTER ONE: INTRODUCTION	1
1.0 Introduction	1
1.1 Problem Statement	1
1.2 Research Questions	2
1.3 Project Objectives	3
1.4 Expected Outcomes	3
1.5 Scope of Work/Research	3
1.6 Significance and Outcome	3
CHAPTER TWO: RELATED WORKS	5
2.0 Existing Syllabi	5
2.1 Industrial Systems	5
2.2 Industrial Internet of Things	7
2.3 Industrial Control	7
2.3.1 Discrete Controllers	8
2.3.2 Distributed Control Systems	10
2.3.3 Supervisory Control and Data Acquisition (SCADA) and Human-Machine Interfaces (HMI)	11
2.3.4 Drives	12
2.3.5 Internet of Things (IoT)	13
2.3.6 Data Analytics	15
CHAPTER THREE: METHODOLOGY	17
3.0 Design Criteria	17
3.1 Comparison of Controllers	18
3.2 Industrial Testbed	19
3.2.1 Aim and Objectives	20
3.2.2 Apparatus	20
3.2.3 Theoretical Background and Methodology	23
3.4 Set of Labs for Syllabus	29
3.4.1 Simple Input and Output Setup	30
3.4.2 Two-Switch PLC Configuration with Latching	31
3.4.3 One-shot Energisation	31

3.4.4	Twenty-Second Contactor Lockout.....	32
CHAPTER FOUR: IMPLEMENTATION AND RESULTS		33
4.0	Labs Implementation and Results	33
4.0.1	Labs	33
4.1	Testbed	37
4.2	IoT Component	38
4.3	Industrial Automation 4.0 Curriculum.....	41
CHAPTER FIVE: CONCLUSIONS AND SUMMARY		43
5.0	Discussion.....	43
5.1	Limitations and Future Work.....	44
APPENDIX.....		45
I.	Syllabi.....	45
II.I	Control Systems.....	45
II.II	Industrial Automation and Control	45
II.	Code.....	48
References.....		56

TABLE OF FIGURES

FIGURE 1: BLOCK DIAGRAM OF THE INDUSTRIAL CONTROL SYSTEM.....	6
FIGURE 2: ARCHITECTURE OF THE IOT NETWORK [16].	14
FIGURE 3: GENERAL OVERVIEW OF INDUSTRIAL AUTOMATION 4.0.....	18
FIGURE 4: MODEL DIAGRAM OF THE TESTBED	19
FIGURE 5: DELTA PROGRAMMABLE LOGIC CONTROLLER	20
FIGURE 6: CHINT CONTACTOR	21
FIGURE 7: AC DRIVE	21
FIGURE 8: THERMOCOUPLE.....	22
FIGURE 9: MOTOR PUMP.....	22
FIGURE 10: ACCELEROMETER IN CONJUNCTION WITH ARDUINO	22
FIGURE 11: A MODEL DIAGRAM SHOWING IOT COMPONENT OF THE TESTBED	24
FIGURE 12: FLOWCHART OF THE TESTBED PROCESS	25
FIGURE 13: IEC 1131-3 [5].....	27
FIGURE 14: BLOCK DIAGRAM OF PLC WIRE CONNECTION.....	30
FIGURE 15: LADDER LOGIC DIAGRAM OF ONE-SWITCH CONFIGURATION LAB.....	30
FIGURE 16: LADDER LOGIC DIAGRAM OF TWO-SWITCH CONFIGURATION.....	31
FIGURE 17: LADDER LOGIC DIAGRAM OF FIVE-SECOND CONFIGURATION	32
FIGURE 18: LADDER LOGIC DIAGRAM OF TWENTY-SECOND LOCK-OUT SETUP	32
FIGURE 19: LADDER LOGIC DIAGRAM OF ONE-SWITCH CONFIGURATION LAB.....	34
FIGURE 20: PHYSICAL CIRCUIT OF LAB ONE	34
FIGURE 21: CIRCUIT DIAGRAM OF LAB CONFIGURATION.....	35
FIGURE 22: LADDER LOGIC DIAGRAM OF TWO-SWITCH CONFIGURATION.....	35
FIGURE 23: LADDER LOGIC DIAGRAM OF FIVE-SECOND CONFIGURATION	36
FIGURE 24: LADDER LOGIC DIAGRAM OF TWENTY-SECOND LOCK-OUT SETUP	37
FIGURE 25: PHYSICAL ARRANGEMENT OF THE TESTBED	38
FIGURE 26: TEST READINGS OF ACCELEROMETER	39
FIGURE 27: SCREENSHOT OF MYSQL DATABASE WITH VALUES	39
FIGURE 28: VALUES DISPLAYED ON A PHP PAGE.....	40
FIGURE 29: GRAPH OF WHEN THE ACCELEROMETER IS AT REST WITH X, Y AND Z-AXIS MOTION REPRESENTED FROM TOP TO BOTTOM RESPECTIVELY.	40
FIGURE 30: GRAPH OF ACCELEROMETER READINGS WHEN IN MOTION.	41
FIGURE 31: GRAPH OF TRANSITION BETWEEN ACCELEROMETER AT REST TO IT IN MOTION AND BACK TO REST	41
FIGURE 32: CODE TO PUSH ACCELEROMETER VALUES ONTO CLOUD	51
FIGURE 33: CODE TO GRAPH ACCELEROMETER VALUES OVER TIME	55

CHAPTER ONE: INTRODUCTION

1.0 Introduction

Industries all over the globe have dominated manufacturing processes and businesses since the 18th century, and it is only natural that the methods of manufacturing over the last 300 years would have advanced. As technology has advanced, ways of acquiring knowledge of controlling manufacturing processes have become less complicated, and challenges initially faced have significantly been reduced. An accompanying problem, however, is that the educational systems of recent times are not always in sync with these technological advances. Thus, this paper looks to build a curriculum for training automation, particularly with Industrial Automation 4.0 and implements a testbed with the key elements of this curriculum. It delves into the existing control systems used in industry namely; the Distributed Control System (DCS), Discrete Controllers and Programmable Logic Controller (PLC) systems and how the Internet of Things (IoT) can be used to enhance industrial automation.

1.1 Problem Statement

Industrial automation 4.0 is the current generation of automation where there is data exchange and also digitization of industrial components. It typically employs the use of cloud computing, IoT and cyber-physical systems. Industrial automation 4.0 creates an environment for faster automation, monitoring of data and data exchange. As an example, Amazon uses such a system in their warehouses to keep track of the stock of items.

Most higher education curricula for majors in Computer Engineering or Electrical Engineering include Industrial Automation or IoT, but hardly incorporate both as one course. This makes it relatively harder for fresh graduates to adapt to the industrial environment since these are hardly taught together. A particular application of this combination is predictive maintenance. Maintenance in the industry is and has always been an important practice, as

equipment and hardware wear out over time. Companies, however, face a conundrum in how frequently they must perform maintenance. Generally, there are two main scenarios with which maintenance is performed. One scenario has the company wait for hardware to break down before performing repairs and maintenance. The other scenario has the company perform maintenance at regular intervals, regardless of whether machinery has broken down or not. Both scenarios obviously have their flaws. A major flaw with the former is that productivity is reduced substantially in waiting for the equipment to be fixed. Another is that it can lead to the complete breakdown of the equipment making it non-operational. The second scenario has its flaw from a more financial perspective. If there is nothing wrong with the machinery, but the company goes ahead with maintenance regardless, the company bears the unnecessary cost. Companies are then left with choosing what would be “the lesser of two evils” where perfectly good components may be replaced before their lifespan ends.

1.2 Research Questions

As stated, industrial control systems have been upgraded over the course of their existence. In an era where technology is as advanced as it has ever been, questions begin to emerge particularly regarding the efficiency of these systems and whether personnel are trained enough to use them. Therefore, this paper seeks to ask the following questions:

- How can IoT be incorporated into the use of industrial control systems?
- How can a set of labs be developed for training in automation?
- How can the maintenance of systems be improved by combining IoT and the training curriculum?
- Can the time for maintenance be predicted more accurately?

1.3 Project Objectives

The project seeks to incorporate the concepts of IoT into industrial control systems. It ultimately aims to create a curriculum based on Control Systems, Industrial Automation and IoT . It also seeks to find an optimal way of performing maintenance on industrial machines in the form of a testbed. As IoT is used in specific systems, this project seeks to analyse stored data from certain systems to improve the systems collectively.

1.4 Expected Outcomes

The key expected deliverables for the project are:

- Proof of concept for integrating IoT and industrial automation.
- An integrated IoT/industrial control system
- End-to-end IoT system including SCADA administrative system.
- Create a curriculum to allow training in industrial automation 4.0.
- Documentation for training

1.5 Scope of Work/Research

The topic of industrial control systems is a rather broad one. Each different control system type, with their features, could interface with a cloud-based system slightly more different than the others would, and that makes this project a large one. Thus, this project will seek to, at the very least, create a testbed representative of the typical components and technologies which comprise industrial automation 4.0.

1.6 Significance and Outcome

Should this be successful, and IoT-based control systems are seen to be more efficient, it could change many things with how companies conduct their manufacturing and control processes. For example, with predictive maintenance, downtime could be monitored and controlled more closely, giving space to increased productivity. More efficient operations

typically signify a cut in long-term cost, which is good news to every money-making entity. A reduction in cost accompanied by the higher quality of products is an ideal scenario for technology advancement and higher living standards. Also, automating particular processes promotes a safer working environment. A disadvantage, however, is that there could be a decrease in employment rate, as automation of processes often means someone has to lose a task to do.

The curriculum would attempt to bridge the gap between the educational and industrial sectors. It could also facilitate an environment where there is enough technical know-how of the current trend of industrial components function. This will provide students with versatility for the Industry as well with knowledge in different fields than would be expected of them at their levels.

CHAPTER TWO: RELATED WORKS

2.0 Existing Syllabi

Industrial automation, though not of the version 4.0, is being taught in undergraduate Electrical Engineering majors. A lot of the concepts of industrial control are also treated with Control System courses. A major flaw with the existing Control Systems syllabi is that they only contain information needed to create the components of these systems theoretically. This means there is only focus on the mathematical modelling of these components. What is actually done or used in industry is rarely elaborated and leaves students with little of what happens in reality. Understandably, a lot of syllabi compensate for this flaw with the Industrial Automation and Control courses. In the Appendix are tables showing the topics under the Control Systems course taken in the University of South Carolina and one in Industrial Automation and Control under the National Programme on Technology Enhanced Learning in India. They both show the flaw this project seeks to address, however. Both courses have no topics which teach the IoT parts of Industry 4.0. Important concepts such as the Internet of Things, Cloud Computing and Data Mining are missing from the syllabus, and this highlights the cause of a significant lack of know-how on how Industry 4.0 works as of now. Instead, these concepts are taught independently as separate courses and not in the context of industrial automation.

2.1 Industrial Systems

An industrial system is a set of machinery which aid to turn raw materials into finished products. They are analogous to mathematical functions in that they take an input, perform an operation on that input and produce an output. Thus, it can be said that industrial systems add value to raw materials. Industrial *control* systems are sets of equipment which, as the name may suggest, control the processing (generally called Industrial Process Control) of the inputs the industrial system receives. Most often, different kinds of machinery are needed to set up industrial control systems, which include sensors, actuators, motors, measuring instruments

and others. Typically, such systems are built around a conveyor belt where all needed components are built around this conveyor belt in fixed locations. One alternative setup, as per Scholl et al. [1], is to make each component wireless to improve flexibility. Scholl's configuration has actuators and sensors communicating wirelessly [1]. Another setup has a smart interconnection between heterogeneous components using an Internet of Things (IoT) system, as per Tao, Cheng and Qi [2]. The IoT is an interconnection of devices and networks to exchange data. Regarding the technologies which have been discussed so far, Figure 1: Block Diagram of the Industrial Control System shows how each component relates to another.

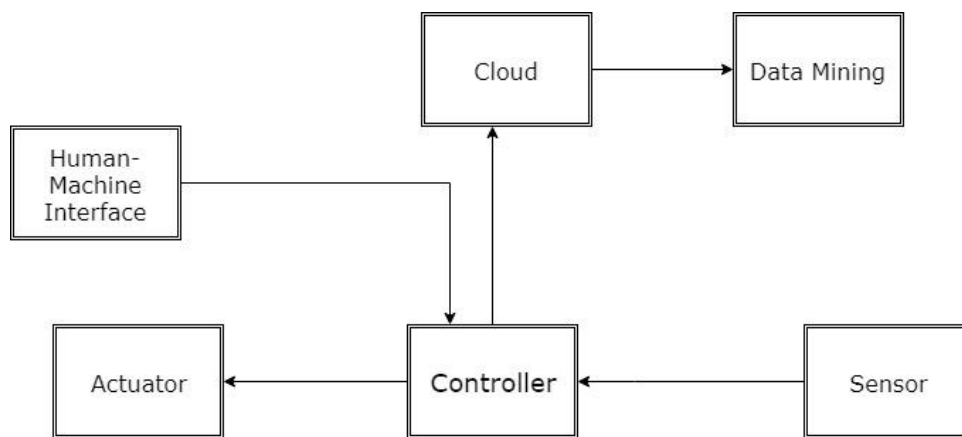


Figure 1: Block Diagram of the Industrial Control System

The diagram shows the typical look of an Industry 4.0 set up. All components are controlled by the controller. The controller and human-machine interface work in tandem to monitor and also control the entire system. This controller, however, works on response to certain values it would receive from the sensor(s) it is connected to. This response, more often than not, would be to move an actuator or piece of machinery. An important part of Industry 4.0 is the presence of IoT which outlines the use of the cloud and data mining resources used. These provide a platform for the analysis of the data the system is provided and how this data could be better than it currently is.

It is critical to also elaborate on the main features of the industrial revolutions which have occurred so far. The first, or Industrial Automation 1.0, included the use of mechanised structures which perform formally manual tasks. Industrial Automation 2.0 introduced the use of assembly lines by using electric structures along with the mechanical structures from Industry 1.0. The third generation performed automation using computers, controllers and mechatronics with devices like robots and Programmable Logic Controllers (PLC). Industry 3.0 is still commonly used, and it's concepts are what is taught in higher education institutions most often.

2.2 Industrial Internet of Things

The Internet of Things is a network of other networks and devices to share information. It is used in a variety of systems like smart homes and smart cities, and it is not rare to see it being used in industrial control systems as well. Therefore, the Industrial Internet of Things (IIoT) is an interconnection of industrial devices such as actuators, sensors and industrial controllers. IIoT has many applications, such as [3]:

- Smart Factories
- Predictive Maintenance
- Smart Wearables
- Smart Logistics Management

2.3 Industrial Control

Industrial control, or process control, uses industrial control systems to optimise processing levels and production through automation. The main components of an industrial control system include controllers, sensors, actuators, drives, Supervisory Control and Data Acquisition (SCADA) systems and timers. There are many different mechanisms with which industrial control is achieved featuring devices like PLCs and mechanisms like PID Control.

Some of the main controllers used in industrial control are discrete controllers, PLCs and Distributed Control Systems.

2.3.1 Discrete Controllers

Discrete controllers are the simplest forms of control systems. Each discrete controller has a single control loop within it. It is for this very reason; they are not exactly the most commonly used control system. Since one typical discrete controller has just one control loop, they usually have one input mapping to one output port. Because of this, experts do not recommend Discrete Controllers for complex systems due to the extra cost and complexity they bring. Discrete controllers also bring the problem of difficulty in scaling up due to their redundant nature. For scaling up, a large number of discrete controllers would be needed which is not ideal.

2.3.1.1 Programmable Logic Controller

Based on the limitations of the generic discrete controller, the Programmable Logic Controller (PLC), was developed. It essentially is a discrete controller on its own, with the ability to support more complex systems. More specifically, they were built to work like and replace the old relay industrial systems due to how cumbersome and costly the latter was [4]. PLCs function by taking input and perform some logical instructions stored in their memory to produce the desired output. PLCs can be thought of as industrial computers optimised for control processes. The PLC is made up of a host of components including a Central Processing Unit (CPU), Memory, Power Supply Unit, Input/output Interface sections, a communication interface and programming device [5]. The CPU is the brain of the system where data processing and instructions are passed and executed while communicating decisions to output devices. The CPU fetches these instructions from the memory where logic is stored to execute certain actions. This logic is placed in the memory using the programming device. The PLC takes data to be processed from an input port and applies the logic in the memory to whatever

devices are connected to the output. All of this receives power from the power supply unit. The voltage rating of the power supply unit typically depends on the type of PLC. However, 24 VDC models are most common. For interaction with other PLCs, a communication interface is vital. It sends and receives data from other PLCs and provide a platform for synchronization with other PLCs. In terms of data buses, the average PLC uses the Hewlett Packard Instrumentation Bus or the General Purpose Instrumentation Bus, as it is known now, for parallel communications. The bus is comprised of 24 lines. 8 of those lines carry data to/from devices connected to the PLC. 5 lines carry data to the PLC's control and status registers. 3 are for handshaking with different devices while the last 8 are ground return lines [5]. PLCs are generally the most preferred controllers due to the ease with which they are programmed. They are designed in a manner which allows engineers without a programming background to easily write programs to them. PLCs are very popular in industrial automation for the following reasons:

- Robustness in their ability to handle adverse conditions like high temperature, vibrations and humidity.
- Interfacing ability with inbuilt input and output ports.
- Readiness to be programmed [5].

2.3.1.1.1 Networking Protocols

The two most common network protocols used in industrial control are Modbus, Process Fieldbus (Profibus) and Fieldbus. These two protocols allow industrial devices to communicate with each other. They establish not only links between controllers but between field devices, actuators and sensors. These protocols are most famous in the use of Distributed Control Systems, where a protocol is needed for the communication between the industrial devices and a remote terminal unit (RTU). Fieldbuses can be defined as local area networks which connect sensors, actuators and controllers [6]. It is with Fieldbus that Profibus is built

upon. It can be considered as a network of Fieldbuses based on a master-slave configuration [7]. Modbus, like Profibus, is a communication protocol used to develop master-slave communications between intelligent devices. Most, if not all, of these protocols, are used to establish communications between controllers and devices to transfer data. It is with these protocols that SCADA systems are able to acquire information from field devices and controllers.

2.3.2 Distributed Control Systems

Distributed Control Systems (DCS) is a collection of unique and independent computing components that monitor particular control variables with a centralised point of control [8]. They are described as a set up where every component receives its instruction from one point. With Industrial IOT adopting a similar definition [8], it makes sense to think of the project objective from this perspective. However, a difference exists between these two systems in how they are coupled with their physical components [8]. Distributed systems are particularly used over wide geographical areas in their operation. Distributed Control Systems typically have four key elements, namely, the controller, local control unit, communication medium and human-machine interface (HMI) [9]. The controller is the supervisory controlling unit over the entire distributed system. The local control unit is used to monitor process variables around field devices and actuators. The human-machine interface may display the trends of the control system to users graphically. Some local HMI units provide control capabilities. However, this is not a very common feature among local control units. Lastly, among the elements, the communication medium consists of media and protocols to make communication between all other elements possible. Particularly for DCSs, two or more communication protocols are required [9]. DCSs are typically used because they manage complex systems better than discrete controllers due to the latter's limited input/output ports.

Along with the extra security they provide, they make systems more scalable [9]. They are typically supplemented with IoT components as they tend to compliment each other. They have their differences, however. DCSs tend to connect to manufacturing equipment and IoT components connect to other devices over a computer network.

2.3.3 Supervisory Control and Data Acquisition (SCADA) and Human-Machine Interfaces (HMI)

SCADA systems are a set of software systems which are used to monitor and control industrial systems from remote locations whilst analysing the systems from the data received. SCADA systems originated from the need to maintain efficiency in industrial systems which were continuously scaling up. Recent industrial systems can stretch into large geographical areas, and a system needed to be created to monitor all components of the system in said areas, thus introducing the SCADA systems [10]. Due to the similar nature of the problems both SCADA and Distributed Control Systems seek to solve, they are used in tandem with one another from time to time. An important advantage of these systems is their ability to acquire data for analysis. Data is typically transferred between a PLC and/or Remote Terminal Unit (RTU) and the SCADA system components which may include operator terminals, computers sensors and field devices [11].

The human-machine interface is often a component of SCADA systems which provides a graphical interaction between the human user of the system and the system itself, as the name may imply. Other names for such a component include a man-machine interface (MMI) and human-computer interface (HCI). Typically, HMIs translate human instructions and inputs into signals machinery would respond to, and vice versa to indicate the response of the machinery. HMIs are used in different electrical and electronic equipment and in large systems like industrial systems, where monitoring of processes could be simplified. Engdahl [12] explains

that the use of an HMI gives the user the advantage of running processes remotely in large industrial areas [12].

2.3.4 Drives

2.3.4.1 AC Drives

AC drives are simply devices which are able to control the speed of electric motors. This is typically done by changing the frequency of the electric motors power supply. Thus, they are more generally called variable frequency drives (VFD). There are three main parts of AC drives. These are:

- Rectifier
- DC Circuit
- Inverter

The rectifier supplies the drive with electrical energy from an electrical network it is connected to. The DC Circuit stores electrical energy coming from the network which would, in turn, be used by the inverter or capacitors. The inverter then uses this stored energy to supply the motor. AC drives are usually classified in two ways; based on the type of AC motor being controlled and based on the type of control functions. Thus, the generally recognised kinds of AC drives are induction, synchronous and variable-reluctance drives [13].

Induction drives are known to be the most common in the industry due to their economic, robust and reliable nature. Essentially, they can be compared to a generic 3-phase transformer with rotating and short-circuited secondary [13]. Synchronous drives are similar to induction drives and can thus, be seen as competitors to the induction drives. A considerable difference is in the way they function. Synchronous drives function by rotating at a synchronous speed. They exhibit proportionality between the frequency they function at and the output speed of

the motor. Variable-reluctance drives show saliency in their motor stators and rotors; that is, the stators and rotors show improved performance in generating power.

2.3.4.2 Variable-Speed Drives

Variable-speed drives (VSD), like AC Drives, control the speed of motors. However, A considerable difference between both of them is what is done to control speed. VSDs control the speed of motors by adjusting the amount of power entering the motors. Therefore, as AC Drives could control only AC devices, the VSD can only control both AC and DC equipment just because it is altering the power reaching the device. VSDs are used in instances where energy needs to be saved, and the speed and torque of process requirements need to be matched [14]. VSDs can be classified into 3 main categories. These are:

- Mechanical variable speed drives
- Electrical variable speed drives
- Hydraulic variable speed drives

Mechanical VSDs work with systems like pulleys and traction systems where mechanical advantage would be needed and are thus classified into two types. These are metallic friction drives and ‘belt and chain’ drives [14]. Electrical VSDs are used most commonly with motors and adapted to work with Eddy currents. Due to their nature of simply adjusting the power going into motors, they are compatible with both AC and DC motors. Barnes [14] goes further to elaborate that hydraulic VSDs are classified into hydrodynamic and hydrostatic types.

2.3.5 Internet of Things (IoT)

The Internet of Things makes use of sensors, cloud computing and computer networks to make the exchange of data between devices and networks possible. It is lauded for its role in increasing performances and saving energy, as well as making the link between devices and their transitioning seamless. For example, in IoT-based homes, called smart homes, there could

be an instance where a coffee machine starts to make coffee when the house owners alarm goes off. The clock and coffee machine exchange data what would be the internet to make such a setup possible.

With IoT being a network, it can be understood that it follows an architecture and certain protocols. The TCP/IP protocol is the most common protocol to be used in computer networks. However, it may not be ideal for IoT networks according to Kraijak and Tuwanut [15]. An IoT network based on a TCP/IP protocol is prone to security and privacy issues due to the possibility of high traffic with the high number of devices that connect to the network. An architecture to address this challenge was developed by Wu et al. [16]. Figure 2: Architecture of the IoT network . shows this architecture.

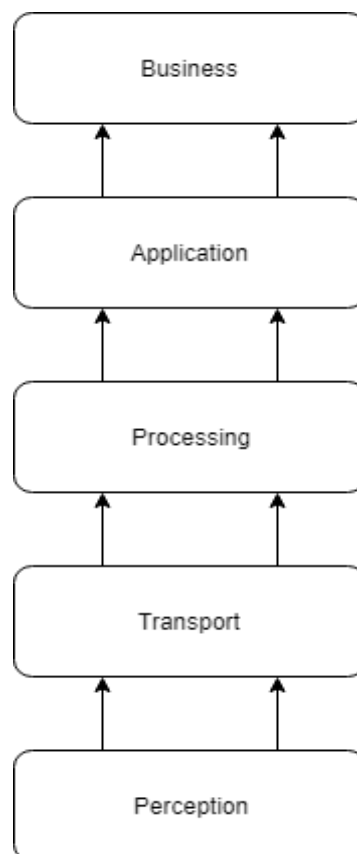


Figure 2: Architecture of the IoT network [16].

The Perception layer handles the general management of physical devices or sensors connected to the network. The Transport layer securely transfers data between devices and a central node

for data processing. Like the name may suggest, the Processing layer stores and processes the data it receives. The Application layer handles the application of processed data. The final Business layer covers the overall management of the IoT application.

The main protocols used in IoT communications include Message Queue Telemetry Transport (MQTT) and Constraint Application Protocol (CoAP) [15]. MQTT is a messaging transport protocol based on one device publishing these messages, and another being subscribed to the former device. The latter device is then alerted when there is a message. CoAP specifies how computationally-constrained devices should function. It allows the simplest and least powerful devices to operate within an IoT network.

2.3.6 Data Analytics

An important component that has emerged from Industry 4.0 is the Data Mining and Analytics component. Data analytics can be described as the science of analysing raw data with the goal of drawing conclusions out of this analysis [17]. A common practice in the industry is to accumulate data from any control process through SCADA systems and perform some form of analytics or refinement to gain knowledge as to what is happening within the process. Trends can then be viewed so as to draw said conclusions. In having this form of information, companies are able to know how productive they are at their current rates and how this can be maximized.

There are four types of Data Analytics. These are descriptive analytics, diagnostic analytics, predictive analytics and prescriptive analytics. Descriptive analytics shows the current trend of happenings within a system. Diagnostic analytics seeks to explain why something may have happened. Predictive analytics uses the data acquired to make prediction towards what would happen if the current trend is to be followed. Finally, prescriptive analytics makes suggestions on what the best course of action may be with respect to some context or

situation [17]. Four major steps are routine in the process of data analytics. These are defined below.

- Determine the group of data and types being worked with
- Collecting this data
- Organization of the data
- Cleaning and filtering of data

Data analytics is used more often than realised even in fields such as marketing to improve performance, and this project seeks to delve into it, albeit, to a scaled-down degree.

CHAPTER THREE: METHODOLOGY

3.0 Design Criteria

The main objective is to build a curriculum which teaches Industrial Automation 4.0. It is meant to result in a testbed which incorporates all the representative automation components. The testbed is meant to be designed in a way which exhibits the most common structures encountered in industry. This is meant to help familiarise students with such systems. It incorporates the use of a computer, controller, a drive, sensors, SCADA setups and microcontrollers. The way these devices work together is outlined below:

- The controller is connected to a piece of equipment as well as data sensor.
- The data sensor detects some quantity related to the equipment it is connected to. For example, for a light bulb, the sensor would detect the light intensity of the bulb continuously and store the data.
- Analytics is done using the computer, SCADA setup and microcontroller on the data found on the cloud. The results of this analytics can be used to quantify performance and optimise processes.

Figure 3: General overview of Industrial Automation 4.0 seeks to represent the flow of the processes is found below.

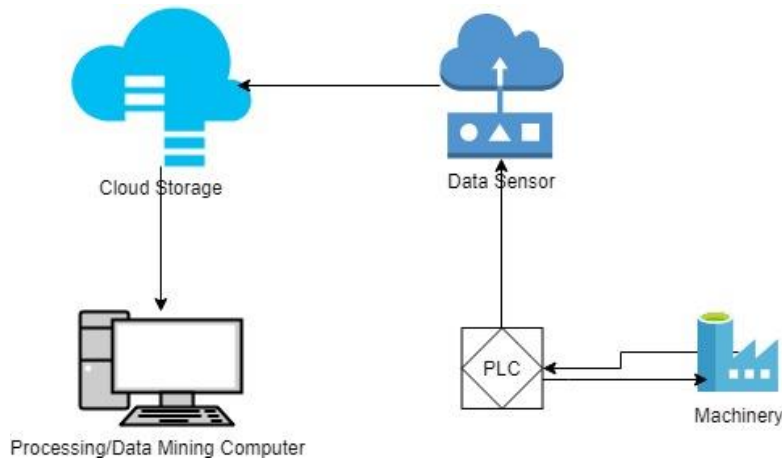


Figure 3: General overview of Industrial Automation 4.0

3.1 Comparison of Controllers

Different controllers are used in industrial automation but are also used for different purposes. It is then necessary to understand the context with which the testbed would be within to compare which controller to use. The main controllers under discussion would be the discrete controller, PLC and DCS. The controllers to be used have already been described and so their areas of use, and advantages over each other would be discussed.

The discrete controller is used in relatively small and temporary systems, particularly when just a singular input and output is needed. In the case of the testbed, many inputs would be required, and so many discrete controllers would be needed. This is not very ideal from a financial context. The DCS is used in large scale systems where the components of a system cannot be monitored manually. The testbed to be implemented is not such a system, and so it would not be needed. The PLC would be the best-suited controller for this testbed. It behaves similarly to a discrete controller but is a more scalable device. Its ability to work with multiple inputs and outputs, as well as its ability to expand the number of these makes it a good fit considering the system.

3.2 Industrial Testbed

Most students have knowledge of Control Theory and some IoT concepts such as Arduino programming. But they know these individually. Therefore, a testbed to combine these fields is described as follows. There would be a need for a tank system which monitors the flow rate of water in the system. The system would include many process variables including temperature and pressure and control the function of a pump as a result of the values of these variables. The system would also have an IoT component to illustrate its importance in Industry 4.0. A model diagram of the testbed can be seen in Figure 4.

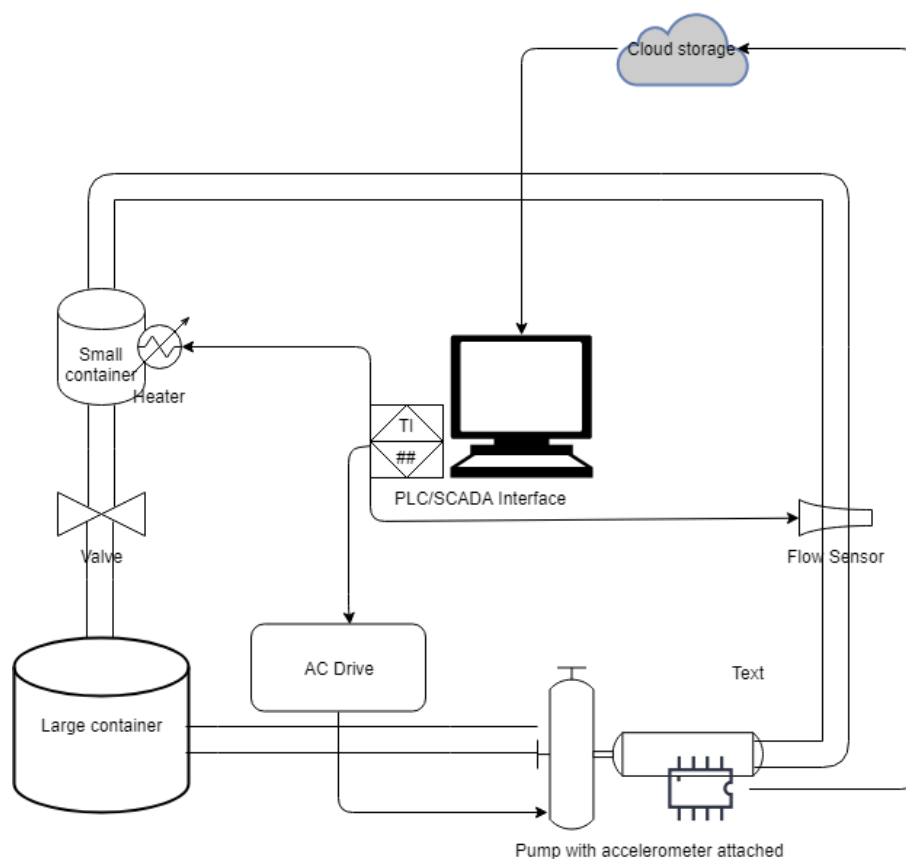


Figure 4: Model diagram of the testbed

3.2.1 Aim and Objectives

The objectives of this testbed generally centre around using industrial control to manage the system. This testbed is meant to show the typical components of a system adopting industrial automation 4.0. The intricacies of this testbed are listed below.

- Pump water when certain depth and temperature conditions are met
- Ability to start and stop the pump using the labs designed for the curriculum
- Monitor flow rate and temperature of the water contained in both storages
- Start or stop AC Drives to control motor speed
- Store vibration data from the system into a cloud
- Analyse data on the cloud to improve this system.

3.2.2 Apparatus

The materials needed for this testbed are mainly industrial components barring the tanks storing the water in the system. These materials are listed below:

- PLC (Delta DVP12SA211R): A programmable logic controller to control the functions of the testbed with sensors.



Figure 5: Delta Programmable Logic Controller

- Contactors: An electrical device for switching circuits

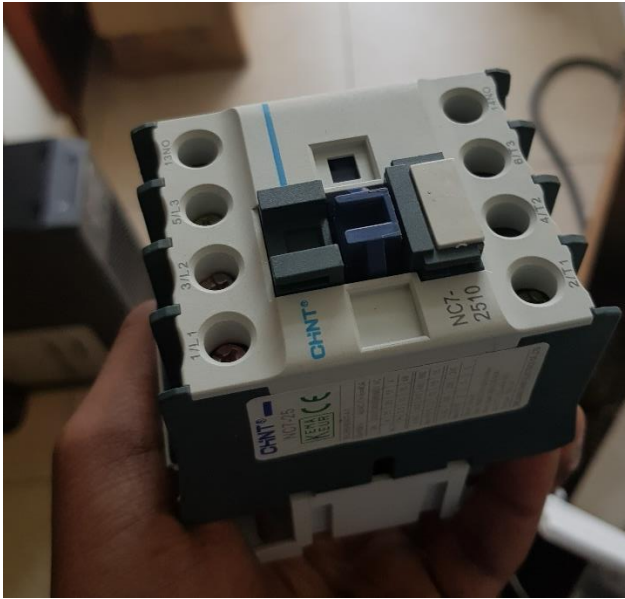


Figure 6: CHiNT Contactor

- 1-inch polyvinyl chloride (PVC) pipes.
- AC Drive: A device which can control the speed of motors



Figure 7: AC Drive

- Ultrasonic level sensor: A sensor which measures the level of water
- Thermocouple: A resistor which measures the temperature of a body via two terminals.

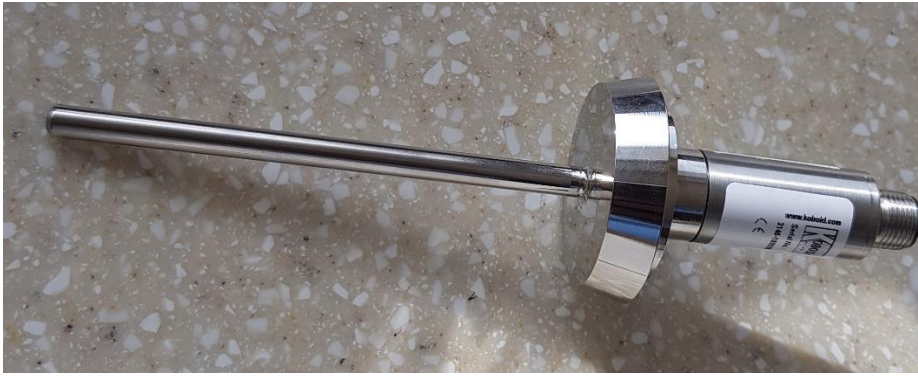


Figure 8: Thermocouple

- Motor Pump: A device which converts electrical energy into mechanical energy



Figure 9: Motor Pump

- ADXL335 accelerometer: Device which shows the acceleration of a body

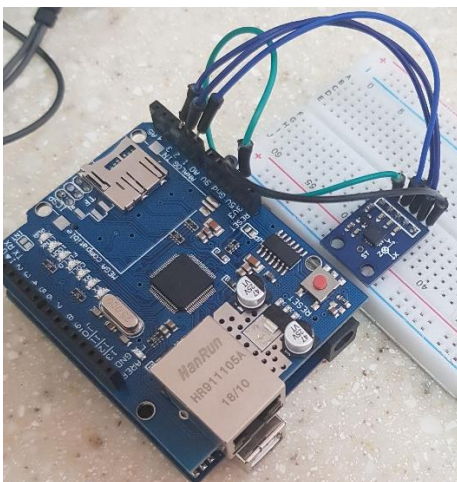


Figure 10: Accelerometer in conjunction with Arduino

- Flow sensor: Sensor for reading the flow within a vessel

3.2.3 Theoretical Background and Methodology

3.2.3.1 Structural Overview

The system is designed to regulate the temperature of the water, as well as the volume of water within a container autonomously. In addition to this, the system must store data onto a cloud for data mining and analysis, to make this system more efficient. The system would have two containers for storing water, one significantly larger than the other. The smaller container would contain the water meant to be sensed with the larger one containing water meant to be fed to the smaller container. These two containers would then be connected to each other via a flow valve. The two containers would flow into each other with the flow rate being monitored by the flow sensor.

The IoT component comes into play by attaching the accelerometer to a pump pushing water from one container to another. Pumps are known to have vibrational patterns which describe how functional the pump is. The accelerometer would be able to help detect if the pump is vibrating abnormally by uploading the sensor values onto a cloud for analysis. This accelerometer would be under the control of an Arduino Uno controller with an ATMEGA328-P microcontroller. Using a SCADA/PLC interface, data could be pulled from the cloud whilst the entire system is being monitored visually. A model diagram illustrating the IoT components can be seen in Figure 11.

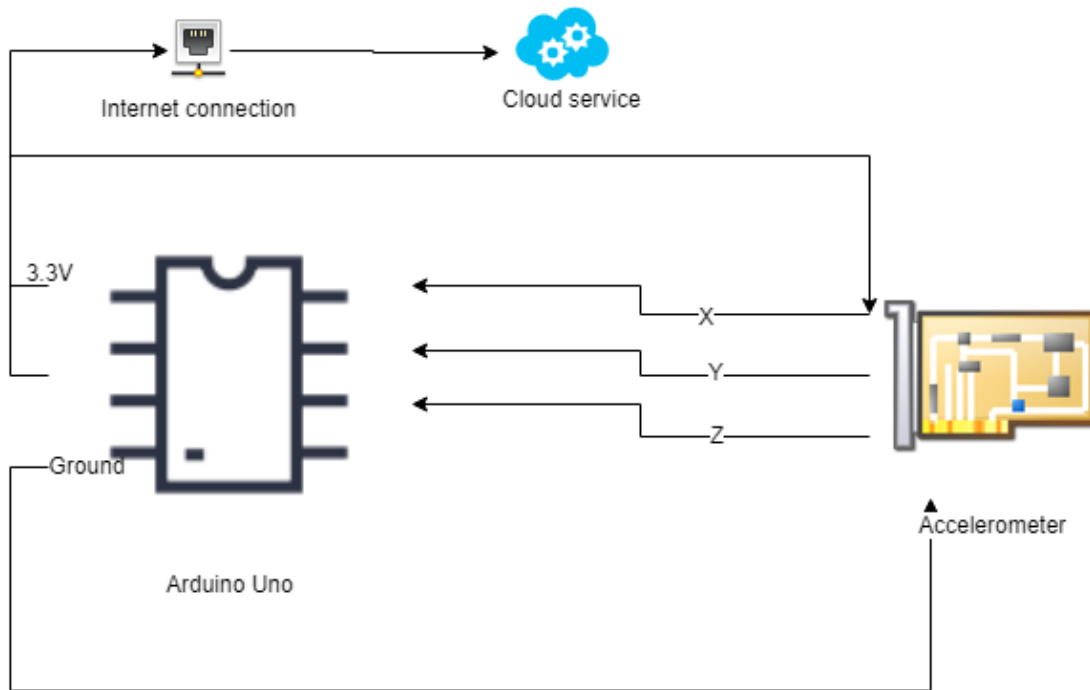


Figure 11: a model diagram showing IoT component of the testbed

3.2.3.2 Workflow of Testbed

The smaller container would hold water along with a level sensor, thermocouple and a heater. The focus of the testbed would be to keep the temperature of the water in the smaller container at a particular value whilst maintaining the level of the water. A flowchart outlining the process can be found in Figure 12 below.

Automation

IoT

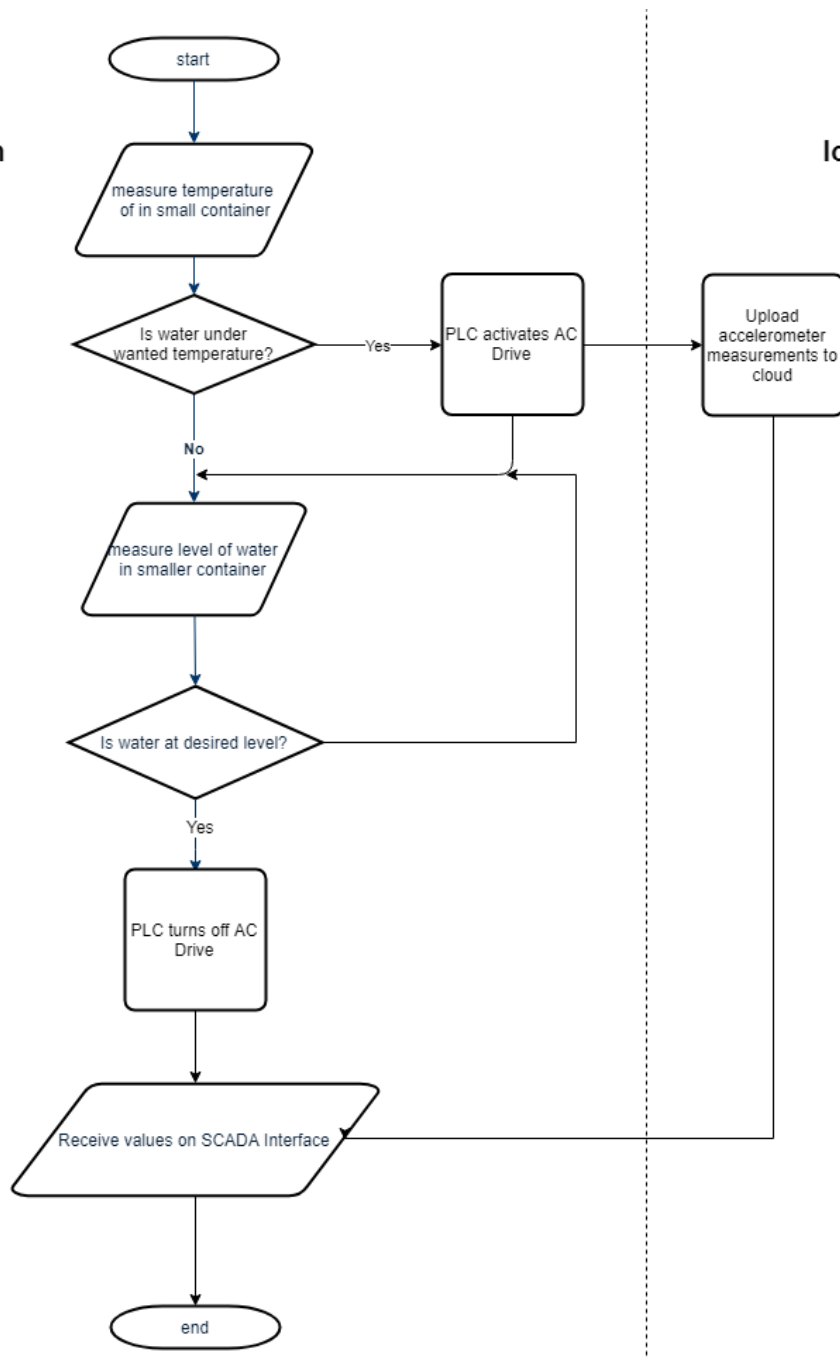


Figure 12: Flowchart of the testbed process

As inferred from the flowchart, the entire process begins by measuring the temperature of the water in the small container. The thermocouple compares the current temperature of the water with an arbitrary setpoint. If the current temperature is less than the setpoint, the PLC activates the heater. If it is the opposite scenario, the PLC activates the AC Drive to pump water from the larger container into the smaller container. The valve into the larger container is also

opened to allow water to flow into it. At the same time, the level sensor and flow sensor communicate with the PLC to detect how much water should be entering the small container to cool the water. If the container reaches the desired level, the PLC turns off the AC Drive. Whilst the pump is active, the accelerometer attached to it records real-time values of its position in space and uploads these values on to a cloud. These values help to deduce the state of the vibration of the pump and can be collected. Data can be analysed to know if bearings are going bad and help deduce when replacements are needed.

3.2.3.3 PID Control

For better accuracy and control, it is imperative that the current values being measured are compared to the setpoint relatively at all points in time. This allows the system to know when it may need to reduce the extent to which it may be performing an action. For example, the testbed must know to what extent the valve must be opened at each point in time because the need for water to run into the large container would differ from time to time. This introduces the concept of Proportional-Integral-Differential (PID) Control. PID is a control mechanism which constantly regulates the error between a setpoint and a process variable and corrects this error using proportional, integral and differential terms. The proportional term provides overall correction action proportional to the error [18]. Thus, smaller errors mean the action would be performed with a lesser extent than with the previous iteration. The integral and derivative terms reduce steady-state errors and improve the transient response of the error correction action respectively [18]. These terms are represented in the equation below as per [19].

$$u(t) = K_P e(t) + K_I \int_0^t e(\tau) d\tau + K_D \frac{d}{dt} e(t) \text{ where}$$

$e(t)$ = error,
 K_P = Proportional gain,
 K_I = Integral gain,
 K_D = Differential gain

With PID Control being very popular amongst industrial machines, it is common to see PLCs which support it. Typically, PLCs make the calculations needed behind the scenes and only take in the setpoint and the gain terms as parameters. The standard symbol used for PID Control in ladder logic is the IEC 1131-1 shown in Figure 13 below.

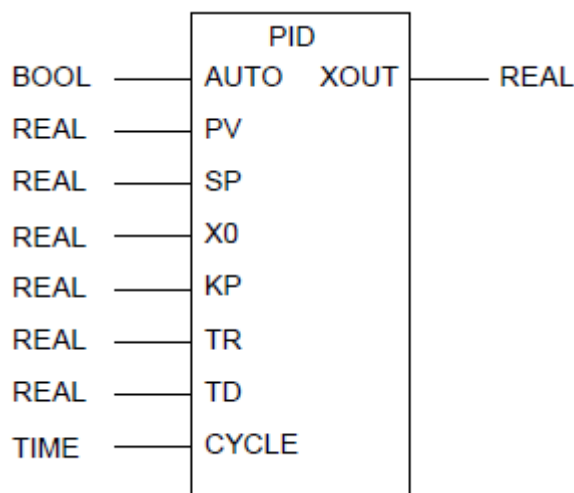


Figure 13: IEC 1131-3 [5]

The pins SP and PV represent the setpoint and process variable respectively. The proportional gain term is represented by KP while TR and TD set the integral and differential time constants respectively, as well. A necessary piece of information is that when AUTO is set to TRUE, the needed adjustments to bring XOUT closer to the setpoint are made, thus autotuning is done to reduce manual adjustments [5]. Thus for this testbench, the PLC would be doing the PID Control and the autotuning to ensure that not too much water is running into the containers or that the AC drive is not powering the motor pump excessively.

3.2.3.4 Analogue I/O

There is the need to factor in the use of analogue values in the testbed. Continuous values in the testbed such as temperature produce analogue voltages when measured. Thus, the PLC provides an avenue for Analogue I/O, Analogue-to-Digital (ADC) conversion and Digital-

to-Analogue (DAC) conversion. PLCs are able to monitor analogue voltages or produce these voltages like for the use of the AC Drive being used in this system [5].

There are different PLC brands to use with the most common being Alan Bradley, Siemens and Delta. However, most PLCs have no or little inputs for analogue values, so they come with extension modules as does the Delta DVP-SA2 PLC model. Accompanying these modules are analogue input specifications. Three decisions must be made to select an analogue input for this testbed.

- Analogue inputs are taken either as unipolar or bipolar values. Unipolar values are positive values within a range of 0 VDC and some positive voltage. However, bipolar values are able to be represented as negative and positive values [5]. Therefore, for this testbed, analogue values would be monitored from the thermocouple, would be bipolar. This is because temperature values can be negative as well. Also, for the PLC to control the AC drive, unipolar values or bipolar would be needed. Thus, a bipolar input system would be considered.
- The range of values would also affect the type of extension module to use. Extension modules are typically deployed (in parts) depending on the range of values they receive with 5 VDC and 10 VDC being the most common. It would then be necessary to use the 10 VDC version considering the AC drive takes in 10 VDC and 4-20 mA to be controlled.
- The third and final characteristic to consider is the resolution of the Analogue/Digital operation. The resolution is the least value of an instrument which can be measured. The manual of the DVP PLC expansion module states that the PLC has 14-bit input ports and 12-bit output ports. The voltage resolution based on this would be the default output voltage divided by 2^n . Therefore, the resolution for the input and output would

0.714 VDC and 0.833 VDC respectively. This helps to factor in measurements in knowing the least values we can possibly measure.

3.4 Set of Labs for Syllabus

For sufficient know-how in the use of Industrial Automation 4.0, it is a necessity to familiarise learners with the most commonly used or most important of these components. Undergraduate courses in Control Systems are typically very mathematical. These courses are most useful in instances where students seek to design their own controllers or control systems, but reliance on this knowledge falls short when working with already designed industrial control systems. Students then do not have sufficient knowledge of how the concepts learnt in this class can be applied to industrial automation systems. They move into the industry without the head start needed to work with the machines available to them. Thus, this section looks at how a set of labs to familiarise undergraduate students with industrial and process control can be designed to supply enough knowledge for the working world. A set of 4 labs were designed which encompassed the more frequently used components. These labs are listed below:

- Simple Input and Output Setup
- Two-Switch PLC Configuration with Latching
- Five-Second Contactor Energisation
- Twenty-Second Contactor Lock-out

A connection from the PLC to a contactor and push button was the more common and most basic connection. Therefore a block diagram to show these connections is seen in Figure 14 below.

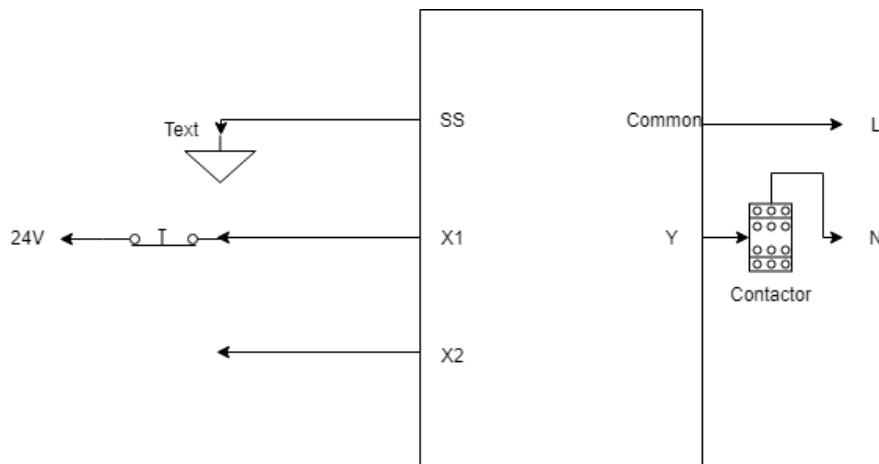


Figure 14: Block diagram of PLC wire connection

3.4.1 Simple Input and Output Setup

The objective of this lab was to familiarise a user of a PLC with ladder diagrams and how a contactor could be opened and closed using a simple push button. It also aims at familiarising the user with the wire connections between the PLC, its power supply, inputs and output devices.

The lab simply requires a setup where the push of a push button connected to a PLC, would close a contactor. Releasing this button would also open the contactor. The PLC performs specific actions based on the instructions uploaded onto it. The ladder logic for this diagram is seen in Figure 15. The ladder diagram shows a straight connection between a normally open (NO) push-button and the output which is the PLC. Pushing the button means the output is activated and that leads to how this should be set up. An application of such a lab would be the operation of a crane motor. The ladder diagram below illustrates how this setup works in Figure 15.



Figure 15: Ladder Logic Diagram of one-switch configuration lab

3.4.2 Two-Switch PLC Configuration with Latching

The aim of this setup was to use two push-buttons to close and open the contactors. Hitting one button was meant to change the state of the contactor indefinitely until the other button had been activated and vice versa.

This setup uses latching in its ladder diagram to achieve its aim. Latching allows parallel connections to the output such that two components don't affect each other until directed to. Figure 7 in the Appendix shows the ladder diagram for this setup. The diagram shows that the latched portion (Y0) only needs the output to be energized once. X0 and X1 then decide the state of the contactor depending on the one which has been activated. This setup is used most generally in instances where a piece of equipment needs to be started. The ladder diagram below illustrates how this setup works in Figure 16.



Figure 16: Ladder Logic Diagram of Two-Switch Configuration

3.4.3 One-shot Energisation

The objective of this lab was to use the PLC's in-built timer to close a contactor for exactly five seconds. It familiarises the learner with the use of the timer and how it can be used.

The set up works as seen in Figure 8 in the Appendix. When X1 is activated, the timer begins to count until it reaches 5 seconds. The instance the timer begins to count, the contactor is closed and opens when there is a time out. This setup can be modified for application in other timing activities. The ladder diagram below illustrates how this setup works in Figure 17.

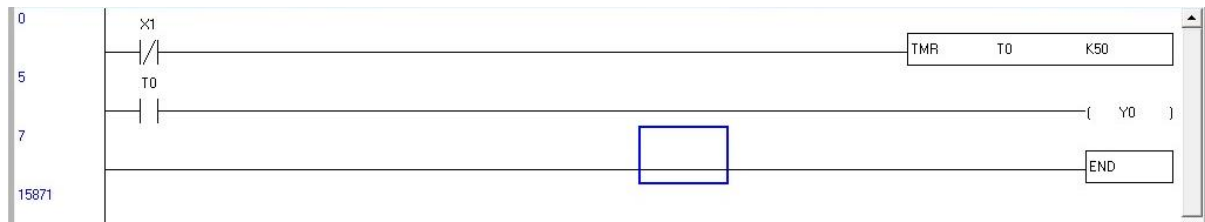


Figure 17: Ladder Logic Diagram of Five-Second Configuration

3.4.4 Twenty-Second Contactor Lockout

This setup aims at locking out a user from closing or opening a contactor after the user consistently does so for 3 times. This kind of mechanism acts as a safeguard against a large flux of power flowing within a device within a short period to prevent overloading/overheating.

This setup uses the PLC's in-built timer and counters to achieve its objectives. It begins the timer when the counter reaches 3. The counter, in turn, is activated every time the two push-buttons are pressed in succession of each other. The ladder diagram for this setup is seen below in Figure 18

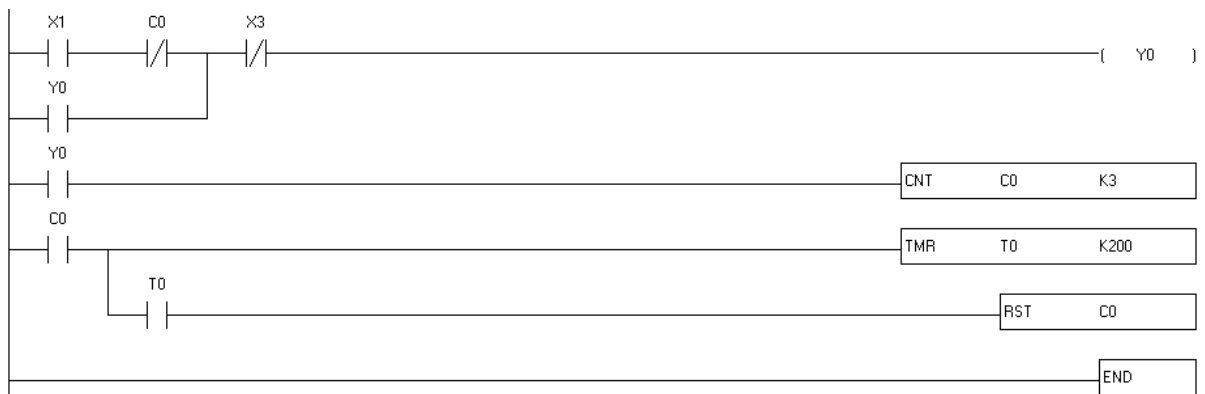


Figure 18: Ladder Logic Diagram of Twenty-Second Lock-out setup

CHAPTER FOUR: IMPLEMENTATION AND RESULTS

4.0 Labs Implementation and Results

4.0.1 Labs

The labs designed were implemented separately and in conjunction with the testbed as well. The results for these labs are seen below.

4.0.1.1 One-Switch Configuration

The first circuit to be wired was the simplest of the collection involving just a push button switch and a contactor with the PLC. The circuit was designed to close the contactor when the switch was pressed down on. When the button is released the contactor will also open accordingly.

The objective of this lab was to activate a 3-phase contactor with a push button using a PLC. As long as the contacts in the push button were closed, the load-bearing contactor is meant to be closed as well. The needed equipment to perform this lab included the following:

- 3-phase contactor
- Push button
- PLC
- Connecting wires.

The theory behind the lab revolves around the use of relay contacts. There are two types of contacts, the normally open (NO) and normally closed (NC) contacts. They can be compared to generic switches such that, closing a contact closes a circuit. Energizing NO contacts close contacts, whilst energizing NC contacts open them, as their names may suggest. This lab is designed to use a NO push-button configuration to close a contactor. The contactor connection included using terminals on the contactor named A1 and A2. These terminals provide a direct

connection to the contactor's coil. One terminal, A1 in this lab, is connected to an output terminal of the PLC and A2 is connected to a neutral pin of a power supply. Delta WSLSoft was used in modelling the ladder logic and the entire lab. The ladder components could be added to a system simply by clicking on their icons or inputting instructions. The diagram below illustrates the ladder logic diagram for the lab.



Figure 19: Ladder Logic Diagram of one-switch configuration lab

Wiring of the entire setup was done by connecting to the common port on the PLC (C0) to live and the S/S port to ground. This created the circuit path for the PLC. The push button was connected to one input port (X1) from one terminal, and a 24V power supply from the other terminal. Likewise, the contactor was connected to an output port (Y0) from one terminal and to neutral from its second terminal. Given the setup, it was expected that pushing the button would close the contactor. Figure 21 illustrates the circuit diagram of this lab and Figure 20 shows the physical connection.

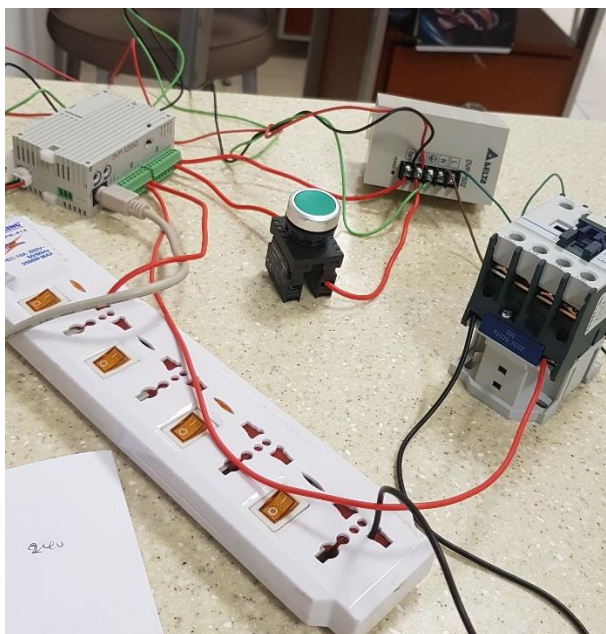


Figure 20: Physical circuit of lab one

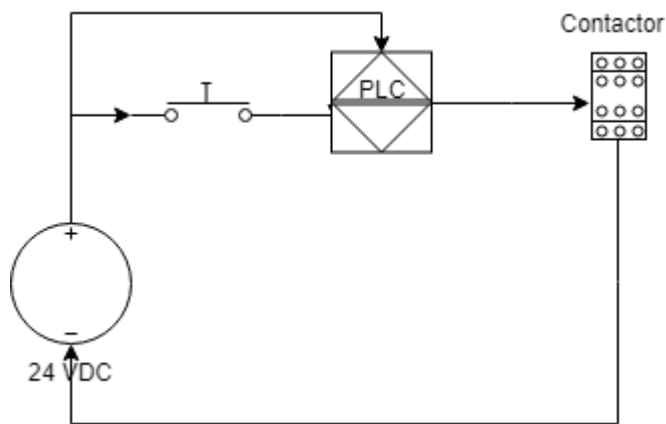


Figure 21: Circuit diagram of lab configuration

4.0.1.2 Two-Switch Configuration

The two-switch configuration was a similar circuit to the one-switch one. Instead, it involved two push button switches, one meant to be “Start” and the other “Stop”. The circuit was designed to have the contactor closed and latched once the Start button had been pressed. The contactor would then stay closed until the Stop button had been pressed. Thus, the only difference in materials needed between this lab and the other is one extra push button.

This configuration made use of ‘latching’ in the ladder logic diagram to make sure the two buttons can be used to close and open the contactor if they are pushed. Latching allows the contactor to remain in a particular state until another input command changes that state. The ladder logic diagram for this configuration is shown below in Figure 22:



Figure 22: Ladder Logic Diagram of Two-Switch Configuration

The latched portion was the NO contact labelled Y0 with the circuit acting in a manner where Y0 being energised energises the Y0 contact to keep it in the state of Y0 continually. X1 represented the Start button, and X3 represented the Stop button. It was expected that pressing the Start button would close the contactor indefinitely until the Stop button is pressed on.

4.0.1.3 One-shot Energisation

The objective of this circuit was designed to keep the contactor closed for a given time period, once a push button had been activated, with the use of the PLC's in-built timer. The period, in this scenario, was five seconds. The needed equipment are the same as in the One-Switch Configuration, and thus, this setup does not require anything new.

This setup uses timers to reach its objective. The ladder logic diagram which represents this circuit is shown below in Figure 23.

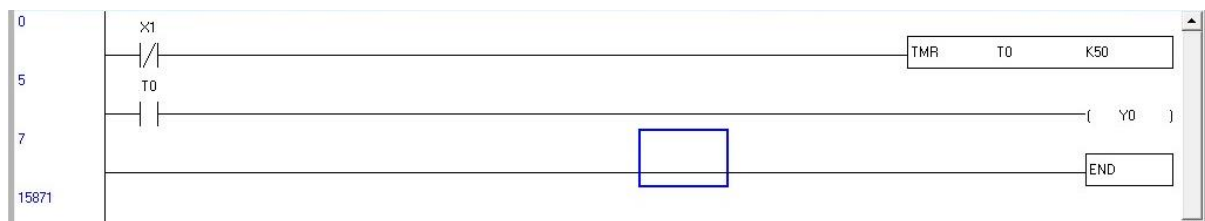


Figure 23: Ladder Logic Diagram of Five-Second Configuration

The timer, represented by T0, is activated when the contact X1 is energised. The timer then counts to 5 seconds, represented by K50, until the contact labelled T0 is energised. Energises T0, in turn, closes the contactor. Given these connections, it is expected that the contactor will close 5 seconds after the button is pushed.

4.0.1.4 Twenty-Second Lock-out

The overall objective of this setup was to allow a user to open and close a contactor not more than 3 times in the space of 20 seconds, with the use of counters and timers. The timer and counter to be used are in-built into the PLC. This kind of setup is particularly meant to protect equipment like motors from sudden fluxes of current in a short time span. This increases the life span of the equipment.

This setup was made possible with coordinated use of the timer and counter. The set up allows the user to close and open the contact three times before starting a 20-second timer,

which prevents the user from doing anything contactor-related. The ladder logic diagram for this setup is found below in Figure 24.



Figure 24: Ladder Logic Diagram of Twenty-Second Lock-out setup

4.1 Testbed

The testbed featured the apparatus listed in the arrangement of the model diagram in Figure 4. It was a culmination of the core ideas of the curriculum and could be described as the result of it. Physically, the structure was seen in Figure 25.



Figure 25: Physical arrangement of the testbed

The figure features the main parts of the testbed barring connections to the AC Drive, PLC, sensors and the smaller container. The testbed was designed to work with the lab setups described in the previous section. Thus, there would be different ways of starting and stopping the motor pump based on the labs described. This testbed is compatible with a SCADA system. The processes can be monitored via the SCADA system and controlled as well in conjunction with the PLC as well.

4.2 IoT Component

The IoT component of the testbed included an accelerometer being controlled by an Arduino Uno board. The values being read by the accelerometer were uploaded onto a database stored on a server and were displayed onto a page. After setting the components up in this way, the accelerometer returned test values as seen in the screenshots below in Figure 26.

```

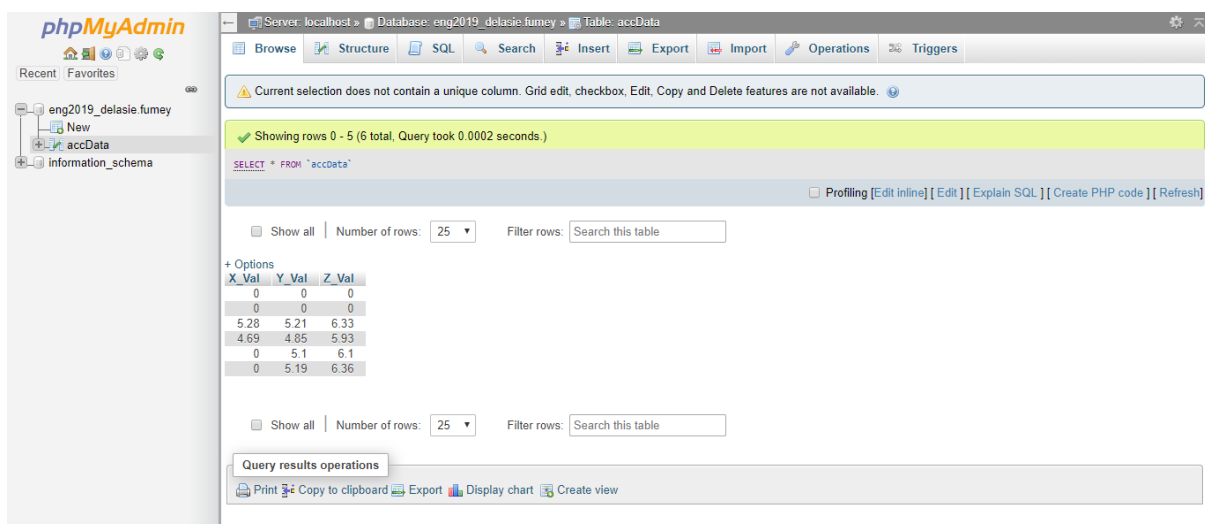
0.00
0.00
0.00
-----
Readings (X, Y, Z):
0.00
0.00
0.00
-----
Readings (X, Y, Z):
0.00
0.00
0.00
-----
Readings (X, Y, Z):
-0.18
-0.31
2.69
-----
Readings (X, Y, Z):
-0.25
-0.18
1.23
-----
Readings (X, Y, Z):
0.34
3.68
2.87
-----
Readings (X, Y, Z):
0.06
-0.01
0.37
-----
Readings (X, Y, Z):
0.18
1.89
2.01

```

☒ Autoscroll ☐ Show timestamp

Figure 26: Test Readings of Accelerometer

Whilst these values are being generated in Arduino's serial monitor, they are also being stored in a MySQL database. The values are then displayed in a PHP page as well. The backend to hold the data used HTTP Get requests to transfer data from the PHP page to the MySQL database. These are requests that retrieve what a webpage displays with respect to parameters provided to the request. Figure 27 and Figure 28 below illustrate these activities.



Server: localhost Database: eng2019_delasie_fumey Table: accData

Current selection does not contain a unique column. Grid edit, checkbox, Edit, Copy and Delete features are not available.

Showing rows 0 - 5 (6 total, Query took 0.0002 seconds)

SELECT * FROM `accData`

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all Number of rows: 25 Filter rows: Search this table

X_Val	Y_Val	Z_Val
0	0	0
0	0	0
5.28	5.21	6.33
4.69	4.85	5.93
0	5.1	6.1
0	5.19	6.36

Show all Number of rows: 25 Filter rows: Search this table

Query results operations

Print Copy to clipboard Export Display chart Create view

Figure 27: Screenshot of MySQL database with values

```

Connected successfully
Error: INSERT INTO accData (X_Val, Y_Val, Z_Val) VALUES (,,)
You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near ',)' at line 1 x value: 0y value: 0z value: 0
x value: 0y value: 0z value: 0
x value: 5.28y value: 5.21z value: 6.33
x value: 4.69y value: 4.85z value: 5.93
x value: 0y value: 5.1z value: 6.1
x value: 0y value: 5.19z value: 6.36

```

Figure 28: Values displayed on a PHP page

The PHP includes an error message due to the fact that no readings were being inserted into the database at that particular moment. The values are graphed using Processing, a piece of software which is most compatible with Arduino outputs and is used for turning the data into meaningful information. The graph shows the pattern of vibration over a course in time. Machine learning algorithms could be run on these graphs to learn what these patterns are. Recommended algorithms would be categorised under unsupervised learning techniques. By doing this, it would be simple to realise when the vibration of the pump is not right. This then helps to inform users of the need to perform some maintenance on the pump. The program is also designed to leave a faded trace of the last readings that were made, and this also makes it simpler to track patterns. Graph readings from test analogue data can be seen below in Figure 29, Figure 30 and Figure 31,

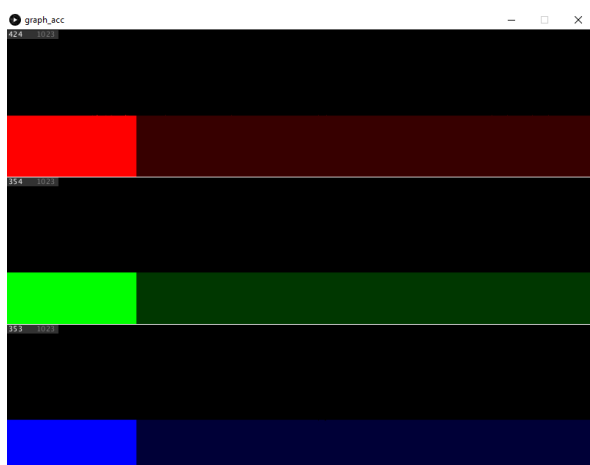


Figure 29: Graph of when the accelerometer is at rest with x, y and z-axis motion represented from top to bottom respectively.

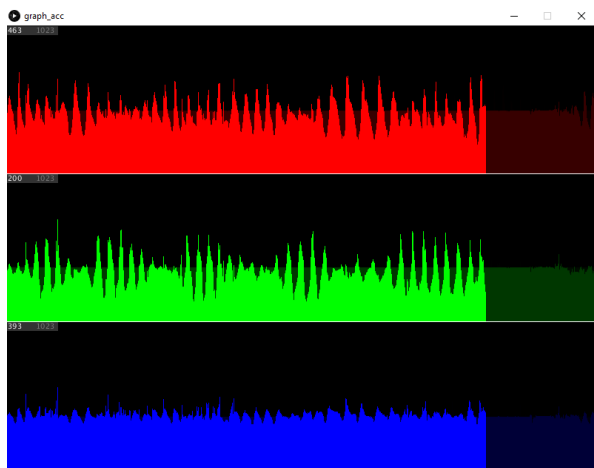


Figure 30: Graph of accelerometer readings when in motion.

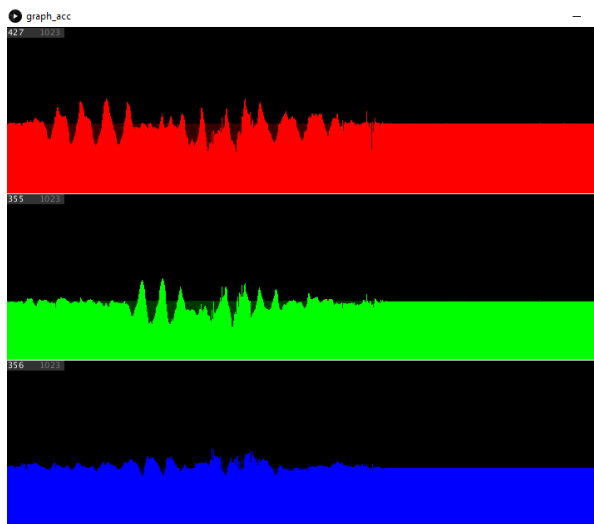


Figure 31: Graph of transition between accelerometer at rest to it in motion and back to rest

4.3 Industrial Automation 4.0 Curriculum

The activities used in creating the curriculum were derived from the implementation of the testbed. The main concepts were used to create a short list of topics which would aid in teaching Industrial Automation 4.0. The syllabus is then tabled below in

Table 1.

Table 1: Updated curriculum

Module No.	Topic
1	Introduction
2	Industrial Systems and Control
3	Instrumentation
4	PID Control
5	Programmable Logic Controllers (PLC)
6	Ladder Logic and PLC Programming
7	PLC Communication Protocols
8	AC Drives
9	SCADA Systems
10	Database Management
11	Web App Development and Technologies
12	Data Analysis and Processing

The curriculum includes topics from the typical undergraduate syllabi with topics such as Industrial Systems and Control, PID Control, PLC Communication Protocols and Ladder Logic and PLC Programming. The new topics are Database Management, Web App Development, Data Analysis and Processing. These topics are important in teaching how to create websites and web apps that store data. It goes further to teach how to analyse and process the data which has been stored.

CHAPTER FIVE: CONCLUSIONS AND SUMMARY

5.0 Discussion

The differentiating factor of Industrial Automation 4.0 from all the other previous generations is the IoT component. This part of the testbed included data from the accelerometer. This data was important towards the implementation of predictive maintenance which is a common application of industrial automation 4.0. The data being observed could be processed, and its pattern could be learnt to find out where there could be disparities. The testbed also included most, if not all of the topics under the curriculum, thus creating a system similar to what is being used in the industry. This gives students the context needed to familiarise themselves with industrial automation 4.0.

The curriculum also attempts to bridge the gap between Control Systems, Industrial Automation and IoT. It combines topics from each of the three courses which would be complementary to all three courses at once. It introduces relatively newer concepts as well in the form of Networks, Communication and Instrumentation. The benefits of such a curriculum range vastly. Not only does it prepare undergraduate students with the sufficient know-how of the recent trend of industrial automation, but also for other parts of the technology community. It provides the versatility which is most sought after among graduates.

With such a testbed, different types of sensors could be used to achieve different functions. This particular testbed is limited in the number of components used but shows the basic input and output functions of a general industrial control system. This representative in the use of the ADXL 335 accelerometer and the PLC, where an output action is done based on specific input actions.

5.1 Limitations and Future Work

Major limitations with the curriculum are mainly in the topics included. Most curricula would include topics covering hydraulic, pneumatic and mechanic systems in general. This curriculum lacks some topics from said fields. They contain concepts used very heavily in manufacturing and factory jobs. They are used in applying mechanisms like conveyor belts, dye casting and compression systems. Nevertheless, these topics would be considered when expanding the curriculum.

APPENDIX

I. Syllabi

II.I Control Systems

Table 2 shows Control Systems course taken at the University of South Carolina. It shows the lack of concepts which would be used in working in an industrial environment.

•	Introduction to Control Systems
•	Laplace Transforms
•	Transfer Function, Stability
•	Block Diagrams and Signal Flow Graphs
•	Physical Systems Model
•	Root Locus Analysis
•	Time Domain Analysis of Control Systems
•	Frequency Domain Analysis of Control Systems
•	Control System Design

Table 2: Syllabus of Control Systems course from the University of South Carolina [20]

II.II Industrial Automation and Control

Table 3 shows a syllabus in Industrial Automation and Control under the National Programme on Technology Enhanced Learning in India. This syllabus lacks the incorporation of IoT components.

Table 3: Industrial Automation Syllabus

Module	Title
I	Introduction
	Architecture of Industrial Automation Systems

II	Measurement Systems Characteristics
	Data Acquisition Systems
III	Introduction to Automatic Control
	PID Control
	PID Control Tuning
	Feed Forward Control Ratio Control
	Time Delay Systems and Inverse Response Systems
	Special Control Structures
	Concluding Lesson on Process Control
	Introduction to Sequence Control, PLC, RLL
	Sequence Control. Scan Cycle, Simple RLL Programs
	Sequence Control. More RLL Elements, RLL Syntax
	A Structured Design Approach to Sequence Control
	PLC Hardware Environment
IV	Flow Control Valves
	Hydraulic Control Systems – I
	Hydraulic Control Systems – II
	Industrial Hydraulic Circuit
	Pneumatic Control Systems – I
	Pneumatic Systems – II
	Energy Savings with Variable Speed Drives
	Introduction to CNC Machines
V	The Field Bus Network – I
	Higher Level Automation Systems

	Course Review and Conclusion
--	------------------------------

II. Code

```
III.  /*
IV.    Web client
V.
VI.    This sketch uploads readings of an adxl335 Accelerometer unto a database
      and
VII.    using an Arduino Wiznet Ethernet shield.
VIII.
IX.    Circuit:
X.      * Ethernet shield attached to pins 10, 11, 12, 13
XI.
XII.    created 18 Dec 2009
XIII.   by David A. Mellis
XIV.    modified 22 Apr 2019
XV.     by Delasie Fumey, based on work by Adrian McEwen
XVI.
XVII.   */
XVIII.
XIX.   #include <SPI.h>
XX.    #include <Ethernet.h>
XXI.
XXII.  // Enter a MAC address for your controller below.
XXIII. // Newer Ethernet shields have a MAC address printed on a sticker on the
      shield
XXIV.  byte mac[] = { 0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED };
XXV.
XXVI.  // if you don't want to use DNS (and reduce your sketch size)
XXVII. // use the numeric IP instead of the name for the server:
XXVIII.      IPAddress server(35,166,18,143); // numeric IP for Google (no DNS)
XXIX.  //char path[] = "/php_connect.php";
XXX.   //char server[] = "http://localhost";    // name address for Google (using
      DNS)
XXXI.
XXXII. // Set the static IP address to use if the DHCP fails to assign
XXXIII.      IPAddress ip(172, 20, 35, 125);
XXXIV.  IPAddress myDns(192, 168, 0, 1);
XXXV.
XXXVI. // Accelerometer initialisers
XXXVII.      int sensorPinX = A2;
XXXVIII.     int sensorPinY = A1;
XXXIX.  int sensorPinZ = A0;
XL.
XLI.    int gyroReadX;
XLII.   int gyroReadY;
XLIII.  int gyroReadZ;
XLIV.
XLV.    int gyroRestX;
XLVI.   int gyroRestY;
XLVII.  int gyroRestZ;
XLVIII.
```



```

XLIX.  double GX;
L.     double GY;
LI.    double GZ;
LII.
LIII.  // Initialize the Ethernet client Library
LIV.   // with the IP address and port of the server
LV.    // that you want to connect to (port 80 is default for HTTP):
LVI.   EthernetClient client;
LVII.
LVIII. // Variables to measure the speed
LIX.   unsigned long beginMicros, endMicros;
LX.    unsigned long byteCount = 0;
LXI.   bool printWebData = true; // set to false for better speed measurement
LXII.
LXIII. void setup() {
LXIV.   // You can use Ethernet.init(pin) to configure the CS pin
LXV.   //Ethernet.init(10); // Most Arduino shields
LXVI.   //Ethernet.init(5); // MKR ETH shield
LXVII.  //Ethernet.init(0); // Teensy 2.0
LXVIII. //Ethernet.init(20); // Teensy++ 2.0
LXIX.   //Ethernet.init(15); // ESP8266 with Adafruit Featherwing Ethernet
LXX.   //Ethernet.init(33); // ESP32 with Adafruit Featherwing Ethernet
LXXI.
LXXII.  // Open serial communications and wait for port to open:
LXXIII.   Serial.begin(9600);
LXXIV.   delay(1000);
LXXV.
LXXVI.  // Calibrate adxl335 to zero at rest
LXXVII.   gyroRestX = analogRead(sensorPinX);
LXXVIII.  gyroRestY = analogRead(sensorPinY);
LXXIX.   gyroRestZ = analogRead(sensorPinZ);
LXXX.
LXXXI.  // Read accelerometer taking values at rest into consideration
LXXXII.   gyroReadX = analogRead(sensorPinX) - gyroRestX;
LXXXIII.  gyroReadY = analogRead(sensorPinY) - gyroRestY;
LXXXIV.  gyroReadZ = analogRead(sensorPinZ) - gyroRestZ;
LXXXV.
LXXXVI.   GX = gyroReadX/67.584;
LXXXVII.  GY = gyroReadY/67.584;
LXXXVIII. GZ = gyroReadZ/67.584;
LXXXIX.
XC.     while (!Serial) {
XCI.    ; // wait for serial port to connect. Needed for native USB port only
XCII.   }
XCIII.
XCIV.   // start the Ethernet connection:
XCV.   Serial.println("Initialize Ethernet with DHCP:");
XCVI.   if (Ethernet.begin(mac) == 0) {
XCVII.   Serial.println("Failed to configure Ethernet using DHCP");
XCVIII.   // Check for Ethernet hardware present
XCIX.   if (Ethernet.hardwareStatus() == EthernetNoHardware) {

```

```

C.      Serial.println("Ethernet shield was not found. Sorry, can't run
        without hardware. :(");
CI.     while (true) {
CII.    delay(1); // do nothing, no point running without Ethernet hardware
CIII.   }
CIV.    }
CV.     if (Ethernet.linkStatus() == LinkOFF) {
CVI.    Serial.println("Ethernet cable is not connected.");
CVII.   }
CVIII.  // try to configure using IP address instead of DHCP:
CIX.    Ethernet.begin(mac, ip, myDns);
CX.     } else {
CXI.    Serial.print(" DHCP assigned IP ");
CXII.   Serial.println(Ethernet.localIP());
CXIII.  }
CXIV.   // give the Ethernet shield a second to initialize:
CXV.   delay(1000);
CXVI.   Serial.print("connecting to ");
CXVII.  Serial.print(server);
CXVIII. Serial.println("...");
CXIX.
CXX.    // if you get a connection, report back via serial:
CXXI.
CXXII.   if (client.connect(server, 80)) {
CXXIII.    Serial.print("connected to ");
CXXIV.    Serial.println(client.remoteIP());
CXXV.
CXXVI.    // while(1){
CXXVII.        // Make a HTTP request to database server:
CXXVIII.    client.print("GET
        /~delasie.fumey/Capstone/php_connect.php?GX=");
CXXIX.    client.print(GX);
CXXX.    client.print("&GY=");
CXXXI.    client.print(GY);
CXXXII.    client.print("&GZ=");
CXXXIII.    client.print(GZ);
CXXXIV.    client.println(" HTTP/1.1");
CXXXV.
CXXXVI.    client.println("Host: eng.ashesi.edu.gh");
CXXXVII.    client.println("Connection: close");
CXXXVIII.    client.println();
CXXXIX.    // }
CXL.    } else {
CXLI.    // if you didn't get a connection to the server:
CXLII.    Serial.println("connection failed");
CXLIII.    }
CXLIV.    beginMicros = micros();
CXLV.    }
CXLVI.
CXLVII.    void loop() {
CXLVIII.

```

```

CXLIX. // if there are incoming bytes available
CL.    // from the server, read them and print them:
CLI.    int len = client.available();
CLII.   if (len > 0) {
CLIII.   byte buffer[80];
CLIV.    if (len > 80) len = 80;
CLV.     client.read(buffer, len);
CLVI.    if (printWebData) {
CLVII.    Serial.write(buffer, len); // show in the serial monitor (slows some
        boards)
CLVIII.   }
CLIX.    byteCount = byteCount + len;
CLX.     }
CLXI.
CLXII. // if the server's disconnected, stop the client:
CLXIII.    if (!client.connected()) {
CLXIV.    endMicros = micros();
CLXV.    Serial.println();
CLXVI.    Serial.println("disconnecting.");
CLXVII.    client.stop();
CLXVIII.    Serial.print("Received ");
CLXIX.    Serial.print(byteCount);
CLXX.    Serial.print(" bytes in ");
CLXXI.    float seconds = (float)(endMicros - beginMicros) / 1000000.0;
CLXXII.    Serial.print(seconds, 4);
CLXXIII.    float rate = (float)byteCount / seconds / 1000.0;
CLXXIV.    Serial.print(", rate = ");
CLXXV.    Serial.print(rate);
CLXXVI.    Serial.print(" kbytes/second");
CLXXVII.    Serial.println();
CLXXVIII.
CLXXIX.    // do nothing forevermore:
CLXXX.    while (true) {
CLXXXI.        delay(1);
CLXXXII.    }
CLXXXIII.    }
CLXXXIV.    }

```

Figure 32: Code to push accelerometer values onto cloud

```

// Code for graphing accelerometer values

// Takes ASCII-encoded strings from serial port and graphs them.
// Expects COMMA or TAB SEPARATED values, followed by a newline, or newline and
carriage return
// Can read 10-bit values from Arduino, 0-1023 (or even higher if you wish)
// Can also read float values

// Last modified April 2019
// by Delasie Fumey based off work of Eric Forman | www.ericforman.com |
//ericjformanteaching.wordpress.com

```

```

import processing.serial.*;
Serial myPort;

int numValues = 3; // number of input values or sensors
// * change this to match how many values your Arduino is sending *

float[] values = new float[numValues];
int[] min = new int[numValues];
int[] max = new int[numValues];
color[] valColor = new color[numValues];

float partH; // partial screen height

int xPos = 0; // horizontal position of the graph
boolean clearScreen = true; // flagged when graph has filled screen

void setup() {
    size(800, 600);
    partH = height / numValues;

    // List all the available serial ports:
    printArray(Serial.list());
    // First port [0] in serial list is usually Arduino, but *check every time*:
    String portName = Serial.list()[0];
    myPort = new Serial(this, portName, 9600);
    // don't generate a serialEvent() until you get a newline character:
    myPort.bufferUntil('\n');

    textSize(10);

    background(0);
    noStroke();

    // initialize:
    // *edit these* to match how many values you are reading, and what colors you
like
    values[0] = 0;
    min[0] = 0;
    max[0] = 1023; // full range example, e.g. any analogRead
    valColor[0] = color(255, 0, 0); // red

    values[1] = 0;
    min[1] = 0;
    max[1] = 1023; // partial range example, e.g. IR distance sensor
    valColor[1] = color(0, 255, 0); // green

    values[2] = 0;
    min[2] = 0;
    max[2] = 1023; // digital input example, e.g. a button

```

```

    valColor[2] = color(0, 0, 255); // blue
    /*
    // example for adding a 4th value:
    values[3] = 0;
    min[3] = 0;
    max[3] = 400; // custom range example
    valColor[3] = color(255, 0, 255); // purple
    */
}

void draw() {
    // in the Arduino website example, everything is done in serialEvent
    // here, data is handled in serialEvent, and drawing is handled in draw()
    // when drawing every loop in draw(), you can see gaps when not updating as fast
    as data comes in
    // when drawing in serialEvent(), you can see frequency of data updates
    reflected in how fast graph moves
    // (either method can work)

    if (clearScreen) {
        // two options for erasing screen, i like the translucent option to see
        "history"
        // erase screen with black:
        //background(0);

        // or, erase screen with translucent black:
        fill(0,200);
        noStroke();
        rect(0,0,width,height);

        clearScreen = false; // reset flag
    }

    for (int i=0; i<numValues; i++) {

        // map to the range of partial screen height:
        float mappedVal = map(values[i], min[i], max[i], 0, partH);

        // draw lines:
        stroke(valColor[i]);
        line(xPos, partH*(i+1), xPos, partH*(i+1) - mappedVal);

        // draw dividing line:
        stroke(255);
        line(0, partH*(i+1), width, partH*(i+1));

        // display values on screen:
        fill(50);
        noStroke();
        rect(0, partH*i+1, 70, 12);
    }
}

```

```

    fill(255);
    text(round(values[i]), 2, partH*i+10);
    fill(125);
    text(max[i], 40, partH*i+10);

    //print(i + ": " + values[i] + "\t"); // <- uncomment this to debug values in
array
    //println("\t"+mappedVal); // <- uncomment this to debug mapped values
}
//println(); // <- uncomment this to read debugged values easier

// increment the graph's horizontal position:
xPos++;
// if at the edge of the screen, go back to the beginning:
if (xPos > width) {
    xPos = 0;
    clearScreen = true;
}
}

void serialEvent(Serial myPort) {
    try {
        // get the ASCII string:
        String inString = myPort.readStringUntil('\n');
        //println("raw: \t" + inString); // <- uncomment this to debug serial input
from Arduino

        if (inString != null) {
            // trim off any whitespace:
            inString = trim(inString);

            // split the string on the delimiters and convert the resulting substrings
into an float array:
            values = float(splitTokens(inString, ", \t")); // delimiter can be comma
space or tab

            // if the array has at least the # of elements as your # of sensors, you
know
            // you got the whole data packet.
            if (values.length >= numValues) {
                /* you can increment xPos here instead of in draw():
                xPos++;
                if (xPos > width) {
                    xPos = 0;
                    clearScreen = true;
                }
                */
            }
        }
    }
}

```

```
}  
catch(RuntimeException e) {  
    // only if there is an error:  
    e.printStackTrace();  
}  
}
```

Figure 33: Code to graph accelerometer values over time

References

- [1] G. Scholl, H.-J. Korber and W. Housam, "Modular Wireless Real-Time Sensor/Actuator Network for Factory Automation Applications," *IEEE Transactions on Industrial Informatics*, pp. 111-119, 2 May 2007.
- [2] F. . Tao, J. . Cheng and Q. Qi, "IIHub: An Industrial Internet-of-Things Hub Toward Smart Manufacturing Based on Cyber-Physical System," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 5, pp. 2271-2280, 2018.
- [3] "Applications of Industrial Internet of Things (IIoT)," RFPAGE, 23 June 2018. [Online]. Available: <https://www.rfpage.com/applications-of-industrial-internet-of-things/>. [Accessed 12 March 2019].
- [4] C. Gonzalez, "Engineering Essentials: What Is a Programmable Logic Controller?," *Machine Design*, 1 June 2015. [Online]. Available: <https://www.machinedesign.com/engineering-essentials/engineering-essentials-what-programmable-logic-controller>. [Accessed 12 November 2018].
- [5] W. Bolton, *Programmable Logic Controllers*, Oxford: Elsevier Newnes, 2006.
- [6] J.-P. Thomesse, "Fieldbus Technology and Industrial Automation," in *2005 IEEE Conference on Emerging Technologies and Factory Automation*, Catania, 2005.
- [7] E. Tovar and F. Vasques, "Real-time Fieldbus Communications Using Profibus Networks," *IEEE Transactions on Industrial Electronics*, vol. 46, no. 6, pp. 1241-1251, 1999.
- [8] K. Iwanicki, "A Distributed Systems Perspective on Industrial IOT," in *2018 IEEE 38th International Conference on Distributed Computing Systems*, Vienna, 2018.
- [9] T. Agarwal, "Everything You Need to Know About Distributed Control System," *El-pro-cus*, [Online]. Available: <https://www.elprocus.com/distributed-control-system-features-and-elements/>. [Accessed November 2018].
- [10] Inductive Automation, "What is SCADA?," Inductive Automation, 12 September 2018. [Online]. Available: <https://inductiveautomation.com/resources/article/what-is-scada>. [Accessed 3 March 2019].
- [11] National Communications System, "Supervisory Control and Data Acquisition," Virginia, 2004.
- [12] J. R. Engdahl, "Walk-through human/machine interface for industrial control". United States of America Patent US6282455B1, 1998.

- [13] B. K. Bose, *Modern Power Electronics and AC Drives*, New Jersey: Prentice Hall PTR, 2002.
- [14] M. Barnes, *Practical Variable Speed Drives and Power Electronics*, Oxford: Newnes, 2003.
- [15] S. Kraijak and P. Tuwanut, "A Survey on IoT Architectures, Protocols, Applications, Securities, Privacy, Real-World Implementation and Future Trends," King Mongkut's Institute of Technology, Bangkok.
- [16] M. Wu, T.-I. Lu, F.-Y. Ling, I. Sun and H.-Y. Du, "Research on the architecture of Internet of things," in *2010 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE)*, Chengdu, 2010.
- [17] J. Frankenfield, "Data Analytics Definition," Investopedia, 25 March 2019. [Online]. Available: <https://www.investopedia.com/terms/d/data-analytics.asp>. [Accessed 14 April 2019].
- [18] K. H. Ang and G. Chong, "PID Control System Analysis, Design and, Technology," *IEEE Transactions on Control Systems Technology*, vol. 13, no. 4, pp. 559-576, July 2005.
- [19] M. J. Mataric, *The Robotics Primer*, Massachusetts: The MIT Press, 2007.
- [20] University of South Carolina , "ELCT 331 - Control Systems," 11 August 2018. [Online].

Available:
https://www.sc.edu/study/colleges_schools/engineering_and_computing/study/electrical_engineering/degree_programs/bachelor_science/331.pdf. [Accessed 12 April 2019].
- [21] A. S. A. Varma, C. V. S. Sarma, P. S. M. Kumar and T. Ravi, "Data Link Control in Data Communication," *IOSR Journal of Electronics and Communication Engineering*, vol. 4, no. 1, pp. 39-47, 2012.
- [22] Tridium, "Jace 8000 Controller," [Online]. Available: <https://www.tridium.com/~media/tridium/enterprisesecurity/documents/jace%208000%20ds%20new%20tridium.ashx?la=en>. [Accessed 20 February 2019].
- [23] S. Tamboli, M. Rawale, R. Thoraiet and S. Agashe, "Implementation of Modbus RTU and Modbus TCP Communication using Siemens S7-1200 PLC for Batch Process," in *2015 International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials (ICSTM)*, Chennai, 2015.
- [24] E. Forman, *Graph Multiple Sensors in Processing source code (Version 1.0)[Source code]*, www.ericforman.com.

[25] D. A. Mellis, *WebClient source code (Version 1.0)[Source code]*, Arduino, 2009.