**DAILY ASSESSMENT FORMAT**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date:** | **30/05/2020** | **Name:** | **DHAMINI C L** |
| **Course:** | **LOGIC DESIGN** | **USN:** | **4AL17EC025** |
| **Topic:** | **Applications of Programmable**  **logic controllers:** | **Semester & Section:** | **6TH & A** |
| **Github Repository:** | **DHAMINI-CL-Course** |  |  |

|  |
| --- |
| **FORENOON SESSION DETAILS** |
|  |
| **Report:**  **Programmable logic controllers [PLC] are computer-based, solid-state, single processor devices that**  **emulate the behavior of an electric ladder diagram. capable of controlling many types of industrial**  **equipment and entire automated systems. The term logic is used because the programming is**  **primarily concerned with implementing logic and switching operations. Input devices such as**  **switches, and output devices such as motors, being controlled are connected to the PLC and then the**  **controller monitors the inputs and outputs according to the machine or process. Originally PLCs**  **were designed as a replacement for hard-wired relay and timer logic control systems. (Hard-wiring**  **means that all of the components were manually connected by wires). PLC consists of two parts i.e.**  **the PLC hardware and programming.**  **The PLC receives information from connected sensors or input devices, processes the data, and**  **triggers outputs based on pre-programmed parameters.**  **Depending on the inputs and outputs, a PLC can monitor and record run-time data such as machine**  **productivity or operating temperature, automatically start and stop processes, generate alarms if a**  **machine malfunctions, and more. PLCs are a flexible and robust control solution, adaptable to almost**  **any application.**  **There are several key features that set PLCs apart from industrial PCs, microcontrollers, and other**  **industrial control solutions:**  **I/O – The PLC’s CPU stores and processes program data, but input and output modules connect the**  **PLC to the rest of the machine; these I/O modules are what provide information to the CPU and**  **trigger specific results. I/O can be either analogue or digital; input devices might include sensors,**  **switches, and meters, while outputs might include relays, lights, valves, and drives. Users can mix**  **and match a PLC’s I/O in order to get the right configuration for their application.**  **Communications – In addition to input and output devices, a PLC might also need to connect with**  **other kinds of systems; for example, users might want to export application data recorded by the PLC**  **to a supervisory control and data acquisition (SCADA) system, which monitors multiple connected**  **devices. PLCs offer a range of ports and communication protocols to ensure that the PLC can**  **communicate with these other systems.**  **Human Machine Interface (HMI) – In order to interact with the PLC in real time, users need an**  **HMI. These operator interfaces can be simple displays, with a text-readout and keypad, or large**  **touchscreen panels more similar to consumer electronics, but either way, they enable users to review**  **and input information to the PLC in real time.**  **PLCs are used for continuously monitoring the input values from sensors and produces the outputs**  **for the operation of actuators based on the program. Every PLC system comprises these three**  **modules:**  **CPU Module**  **A CPU module consists of central processor and its memory. The processor is responsible for**  **performing all the necessary computations and processing of data by accepting the inputs and**  **producing the appropriate outputs.**  **Power Supply Module**  **This module supplies the required power to the whole system by converting the available AC power**  **to DC power required for the CPU and I/O modules. The 5V DC output drives the computer**  **circuitry.**  **I/O Modules**  **The input and out modules of the programmable logic controller are used to connect the sensors and**  **actuators to the system to sense the various parameters such as temperature, pressure and flow, etc.**  **These I/O modules are of two types: digital or analogue.**  **Communication Interface Modules**  **These are intelligent I/O modules which transfers the information between a CPU and**  **communication network. These communication modules are used for communicating with other**  **PLC’s and computers, which are placed at remote place or far-off locate.**  **The program in the CPU of programmable logic controller consists of operating system and user**  **programs. The purpose of the operating system with CPU is to deal with the tasks and operations of**  **the PLC such as starting and stopping operations, storage area and communication management, etc.**  **A user program is used by the user for finishing and controlling the tasks in automation.**  **Programming A PLC**  **In these modern times, a PC with specially dedicated software from the PLC manufacturer is used to**  **program a PLC.  The most widely used form of programming is called ladder logic. Ladder logic**  **uses symbols, instead of words, to emulate the real world relay logic control. These symbols are**  **interconnected by lines to indicate the flow of current through relay like contacts and coils. Over the**  **years the number of symbols has increased to provide a high level of functionality.**  **The completed program looks like a ladder but in actuality it represents an electrical circuit. The left**  **and right rails indicate the positive and ground of a power supply. The rungs represent the wiring**  **between the different components which in the case of a PLC are all in the virtual world of the CPU.**  **So if you can understand how basic electrical circuits work then you can understand ladder logic.**  **Today, a number of different programming languages are used, but each PLC supplier has their own**  **programming specifications based on the IEC 61131-3 standard. Although they have roughly the**  **same sort of components found in many other computer systems, PLCs operate quite differently. A**  **PLC operating cycle, or scan, consists of:**  ** Reading and storing the current value of each input,**  ** Changing all physical outputs to match the output table values stored in data memory,**  ** Sequentially executing the instructions in program memory, while storing any updated variables or**  **outputs to data memory.** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Date:30/5/2020** |  | **Name: DHAMINI C L** |  | |
| **Course:PYTHON** |  | **USN:4AL17EC025** |  | |
|  |  |  |  | |
| **Topic: mobile app** |  | **Semester & Section:6TH A SEC** |  | |
| **AFTERNOON SESSION DETAILS** | | | |
| **Image of session** | | | |
| **Report – Report can be typed or hand written for up to two pages.**  The overall process for creating a package for IOS can be explained in 4 steps:   1. [Compile the distribution](https://kivy.org/doc/stable/guide/packaging-ios.html#compile-the-distribution) (python + modules for IOS) 2. [Create an Xcode project](https://kivy.org/doc/stable/guide/packaging-ios.html#create-an-xcode-project) (and link your source code) 3. [Update the Xcode project](https://kivy.org/doc/stable/guide/packaging-ios.html#update-the-xcode-project) 4. [Customize the Xcode project](https://kivy.org/doc/stable/guide/packaging-ios.html#customize)  Prerequisites You need to install some dependencies, like Cython, autotools, etc. We encourage you to use [Homebrew](http://mxcl.github.com/homebrew/) to install those dependencies:  brew install autoconf automake libtool pkg-config  brew link libtool  sudo easy\_install pip  sudo pip install Cython==0.29.10  For more detail, see [IOS Prerequisites](https://kivy.org/doc/stable/guide/packaging-ios-prerequisites.html#packaging-ios-prerequisites). Just ensure that everything is ok before starting the second step! Compile the distribution Open a terminal, and type:  $ git clone git://github.com/kivy/kivy-ios  $ cd kivy-ios  $ ./toolchain.py build kivy  Most of the python distribution is packed into python27.zip. If you experience any issues, please refer to our [user group](https://groups.google.com/forum/#!forum/kivy-users) or the [kivy-ios project page](https://github.com/kivy/kivy-ios). Create an Xcode project Before proceeding to the next step, ensure your application entry point is a file named main.py.  We provide a script that creates an initial Xcode project to start with. In the command line below, replace test with your project name. It must be a name without any spaces or illegal characters:  $ ./toolchain.py create <title> <app\_directory>  $ ./toolchain.py create Touchtracer ~/code/kivy/examples/demo/touchtracer  Note  You must use a fully qualified path to your application directory.  A directory named <title>-ios will be created, with an Xcode project in it. You can open the Xcode project:  $ open touchtracer-ios/touchtracer.xcodeproj  Then click on Play, and enjoy.  Note  Everytime you press Play, your application directory will be synced to the <title>-ios/YourApp directory. Don’t make changes in the -ios directory directly. Update the Xcode project Let’s say you want to add numpy to your project but you did not compile it prior to creating your XCode project. First, ensure it is built:  $ ./toolchain.py build numpy  Then, update your Xcode project:  $ ./toolchain.py update touchtracer-ios  All the libraries / frameworks necessary to run all the compiled recipes will be added to your Xcode project. Customize the Xcode project There are various ways to customize and configure your app. Please refer to the [kivy-ios](http://www.github.com/kivy/kivy-ios) documentation for more information. Known issues All known issues with packaging for iOS are currently tracked on our [issues](https://github.com/kivy/kivy-ios/issues) page. If you encounter an issue specific to packaging for iOS that isn’t listed there, please feel free to file a new issue, and we will get back to you on it.  While most are too technical to be written here, one important known issue is that removing some libraries (e.g. SDL\_Mixer for audio) is currently not possible because the kivy project requires it. We will fix this and others in future versions. FAQApplication quit abnormally! By default, all the print statements to the console and files are ignored. If you have an issue when running your application, you can activate the log by commenting out this line in main.m:  putenv**(**"KIVY\_NO\_CONSOLELOG=1"**);**  Then you should see all the Kivy logging on the Xcode console. How can Apple accept a python app We managed to merge the app binary with all the libraries into a single binary, called libpython. This means all binary modules are loaded beforehand, so nothing is dynamically loaded. | | | |