FarmBot Simulator: towards a Virtual Environment for Scaled Precision Agriculture

Victor Alexander Murcia, Juan Felipe Palacios, Giacomo Barbieri

Mechanical Engineering Department, Universidad de los Andes, Bogotá (Colombia)
{va.murcia10, jf.palacios, g.barbieri}@uniandes.edu.co

Abstract. Precision agriculture (PA) is considered the future of agriculture since allows optimizing returns on inputs while preserving resources. To investigate PA strategies, test-benches can be utilized for reproducing the use of the technology in scaled farms before its implementation. The FarmBot CNC robot farming machine has the premises to become a test-bench for PA, but its control software must be adapted for implementing farming strategies based on the feedback of sensors. In line with the Virtual Commissioning technology, a FambBot Simulator is proposed in this paper for validating the FarmBot control software before its deployment in the physical test-bench. A case study is implemented for demonstrating that the developed FarmBot Simulator allows the verification of different PA strategies through the implementation of a Software-in-the-Loop Simulation. Due to the importance of food security in the 'worldwide landscape', we hope that the FarmBot approach may become a future strategy for the investigation in PA due to its low-cost implementation and its open-source mission.

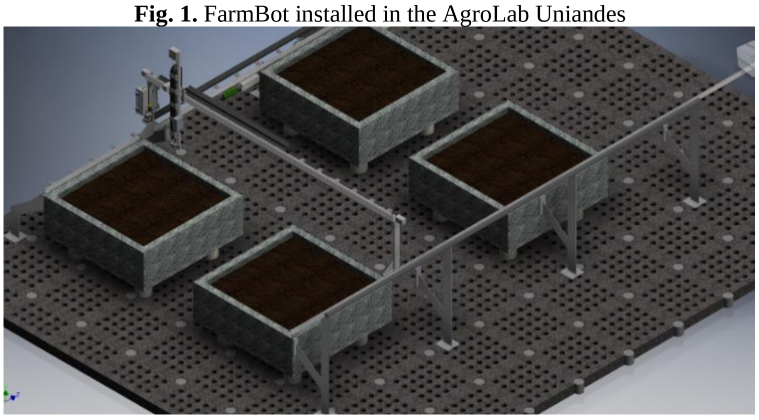
Keywords: Society 5.0, Precision Agriculture, Soil-based Agriculture, FarmBot, Virtual Commissioning.

1 Introduction

Society 5.0 is defined as "A human-centered society that balances economic advancement with the resolution of social problems by a system that highly integrates cyber and physical space" [1]. *Food security* is considered as a current and future social problem since global population is exceeding 7.2 billion and is continuously increasing. In 2050, population has been estimated to reach 9.6 billion [2], and productivity needs to be almost 50% more than that of 2012 [3].
Traditional *soil-based agriculture* allows producing large amount of food but with a negative impact on the environment [4]. It is estimated that 87% of the yearly consumed freshwater is utilized for agriculture [5], and high percentages of water and fertilizer are lost in the ground due to leaching [6]. Traditional agricultural systems are not sustainable on a long run and farmers are under pressure to reduce or eliminate nutrient-laden water discharges to the environment.
Precision agriculture (PA) is a farming management concept based on observing, measuring, and responding to inter and intra-field variability in crops. The goal of PA research is to define a *decision support system* for the whole farm management with the

objective of optimizing returns on inputs while preserving resources [7]. PA has been enabled by the integration of GPS with technologies of the *digital transformation* such as real-time Sensors, Internet of Things, Big Data and Artificial Intelligence. The farmer's and/or researcher's ability to locate the crop precise position in the field allows for the creation of maps able to represent the *spatial variability* of as many variables as can be measured. Variable rate technology (e.g. seeders, sprayers, etc.) uses this data in conjunction with satellite imagery to optimally distribute resources.
In the context of PA, *test-benches* can be used for reproducing scaled farms to investigate the use of technology before its implementation. The *AgroLab Uniandes* was founded in 2019 with this objective and contains different production systems spacing from traditional agriculture to hydroponics and aquaponics [8]. Among the available technologies, a FarmBot was implemented for investigating PA strategies in soil-based agriculture.
The *FarmBot project* developed an open-source precision agriculture CNC robot farming machine [9]. The *FarmBot Uniandes* was mounted on the top of four pots for concurrently investigating different farming strategies; see Figure 1. In this way, different strategies can be implemented since the crops are physically separated in four clusters. The FarmBot developers designed a '*FarmBot Operating System*' that runs on a Raspberry Pi controller. The Raspberry Pi communicates with the '*Farm Design*' web application which allows the user to configure the layout of the scaled farm and to schedule *farming operations* at a specific day and time. However, in the current version of the 'FarmBot Operating System', farming operations cannot be controlled based on the feedback of sensors. Therefore, a different *control software* must be developed to convert the FarmBot into a test-bench for PA.
In the industrial automation domain, *Virtual Commissioning* technology is used to virtually validate the code before its deployment [10]. Therefore, the objective of this work is to develop a *FarmBot Simulator* for designing PA strategies before its implementation in the physical test-bench. The paper is structured in this way: Virtual Commissioning is summarized in Section 2, while the proposed FarmBot Simulator is described in Section 3. Section 4 uses the simulator for the design of different PA strategies. Obtained results are discussed in Section 5, while Section 6 presents the conclusions and sets the directions for future work.

Fig. 1. FarmBot installed in the AgroLab Uniandes



2 Virtual Commissioning

Innovation is defined as the application of better solutions that meet new requirements, unarticulated needs, or existing market needs [11]. Although a universally framework for identifying all the types of innovation is not available, few categories are generally utilized for describing an innovation such as product/process innovation and disruptive/incremental/radical/architectural innovation [12].

Process innovations are innovations in the way an organization conducts its business, such as in the techniques of producing or marketing goods or services. *Architectural innovations* are innovations that result from the application of existing technology to a different market. In this work, we propose to apply a widespread technology of the industrial automation domain for improving the process of designing and verifying control strategies for a scaled PA application. Therefore, the presented solution can be defined as a process and architectural innovation.

Virtual Commissioning (VC) consists in the early development and validation of control software using a simulation model [13]. VC is performed by connecting a simulation model that reproduces the behavior of the physical plant to the hardware or emulated controller which contains the software to be validated. VC requires the *virtual plant model* to be fully described at the level of sensors and actuators.

VC enables the evaluation of the system's functionality, performance and safety before its physical assembly and commissioning [10]. Nowadays, VC is implemented in many *automation applications* [14], due to its ability to speed up-the commissioning process. Without VC, the production system must be stabilized solely by real commissioning with real physical plants and real controllers, which is expensive and time consuming. Whereas, an industrial study showed that real commissioning time is reduced approximately by 75% when VC is implemented beforehand [15].

As shown in [10], there can be two possible configurations in VC: (i) *Hardware-in-the-Loop* (HiL) commissioning involving a virtual plant and a real controller; (ii) *Software-in-the-Loop* (SiL) commissioning involving a virtual plant and a virtual controller. HiL commissioning is generally utilized for validating hard real-time applications, while SiL commissioning for soft real-time ones. Since the FarmBot does not need to fulfill hard real-time requirements, *Software-in-the-Loop Commissioning* is proposed within this work.

Eventually, the following *requirements* must be fulfilled to develop a simulator for SiL commissioning in accordance with the Farmbot project mission and able to verify different PA strategies:

- *Open-source software*: since the 'Farmbot project' is defined as an open-source precision agriculture CNC farming project, the FarmBot Simulator must run into open-source software
- *Interface*: SiL commissioning is valuable only if the verified control software can be deployed into the controller without the need of changes, which may constitute a source of error. Therefore, the developed simulator must implement the same interface of the 'FarmBot Operating System' to enable the seamless deployment of the verified code

- *Movement and irrigation*: only these two functionalities will be implemented in the first version of the simulator. The reason is to obtain an early feedback from the scientific community before the development of additional functionalities
- *Graphical display*: a graphical display able to represent the system state must be available to facilitate the debug operation.

3 FarmBot Simulator

In this section, the developed FarmBot Simulator for designing PA strategies is presented. The architecture implemented for the SiL commissioning is shown in Section 3.1, while Section 3.2 describes the software selected for the SiL simulation. Eventually, the FarmBot Simulator is illustrated in Section 3.3.

3.1 Software-in-the-Loop Commissioning

The FarmBot control architecture is illustrated in the upper part of Figure 2 (physical domain). The *FarmBot* sensors and actuators are controlled through a Farduino board[1] programmed with an Arduino MEGA 2560 microcontroller. The *Farmduino* acts as a middleware interfacing the Farmbot I/Os with the Raspberry Pi microcontroller. The *Raspberry Pi* constitutes the 'brain' of the system since contains the control software and connects the robot to the '*Farm Design*' web application through a MQTT Gateway[2]. FarmBot's Raspberry Pi runs a custom operating system named '*FarmBot OS*' to execute scheduled events and upload sensor data. The OS communicates with the Farmduino over a *serial connection* to send G- and F-code commands, and receive collected data through R- and Q- code. G-code is the code used in numerical control programming, while F-, R- and Q-code contain custom functions created from the FarmBot developers[3]. In the current version of the 'FarmBot OS', farming operations are scheduled through the 'Farm Design' web application and cannot be controlled based on the feedback of sensors. Therefore, a different *control software* must be developed to convert the FarmBot into a test-bench for PA.
Since the Farmduino code will be left unchanged, the *FarmBot Simulator* must implement the Raspberry-Farmduino interface; see lower part of Figure 2 (cyber domain). In this way, the control software can be designed and validated using the simulator, and seamlessly deployed into the FarmBot's Raspberry Pi.

3.2 Software-in-the-Loop Simulation

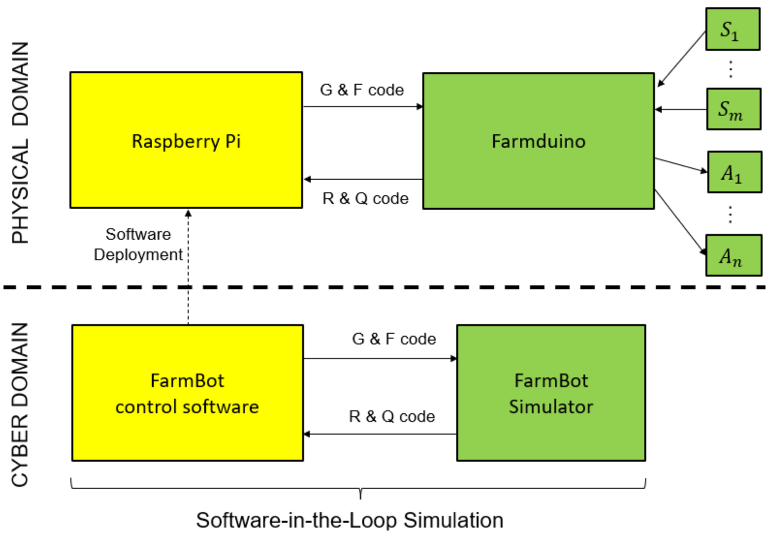
Since the 'Farmbot project' is defined as an open-source precision agriculture CNC farming project, the SiL simulation must be implemented with open-source software. In order to allow the seamless deployment of the verified code, the 'FarmBot control

1 https://farm.bot/products/v1-5-farmduino
2 https://software.farm.bot/docs/intro
3 https://github.com/FarmBot/farmbot-arduino-firmware

software' and the FarmBot Simulator must communicate through a serial port and exchange functions written with the G-, F-, R- and Q-code illustrated in Section 3.1.
The chain of software selected for the SiL simulation is shown in Figure 3 and consists of:

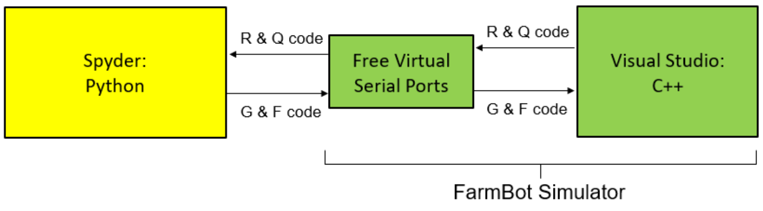
- *Spyder*[4]: open source cross-platform integrated development environment (IDE) for scientific programming in the Python language. This IDE can be installed on Raspberry PI allowing the deployment of the control software verified with the SiL simulation
- *Visual Studio*[5]*:* IDE provided from Microsoft®. Visual Studio supports 36 programming languages and its most basic edition, the Community edition, is available free of charge. Since Farmduino is programmed in C++, this programming language is selected for the FarmBot Simulator. In this way, most of the functions provided from the FarmBot developers can be reused for building the FarmBot Simulator
- *Free Virtual Serial Ports*[6]*:* Windows® application which allows the user to create software virtual serial ports emulating the behavior of physical serial ports. Through this software, Spyder and Visual Studio are interfaced with a virtual serial port. This interface enables the seamless deployment of the verified control software since the communication functions are the same for both the cyber and the physical domain.

Fig. 2. Architecture implemented for the Software-in-the-Loop commissioning



4 https://www.spyder-ide.org/
5 https://visualstudio.microsoft.com/
6 https://freevirtualserialports.com/

Fig. 3. Software selected for the Software-in-the-Loop simulation



3.3 FarmBot Simulator

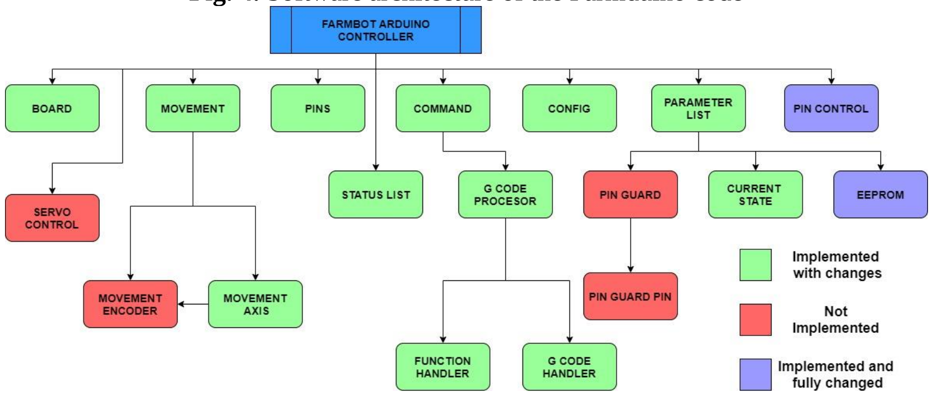
The *Farmbot simulator* was built adapting the classes provided from the Farmduino controller. Figure 4 illustrates a hierarchical representation of the *Farmduino classes*. The classes implemented with changes are shown in green, the ones completely changed in blue, while the ones not implemented in red. Changes were performed to adapt the Farmduino classes to run in the virtual environment. However, the FarmBot Simulator presents the same interface of the Farmduino controller, making the seamless deployment of the verified code viable.
Next, a description of the Farmduino classes is presented (Fig. 4):

- *Farmbot Arduino Controller*: contains the setup and loop functions and constitutes the main class that instantiates all the others
- *Board:* defines the version of the Farmbot PCB board
- *Pins:* configures the physical pins of the ATMEGA 2560 controller
- *PinControl:* controls the pins of the ATMEGA 2560. In our simulator, we replace this class with a new one called "*Arduino Pins"*. The implemented class virtualizes each physical pin storing its commanded value, the 'pin mode' (i.e. Input/Output) and the 'pin type' (i.e. Analog/Digital)
- *PinGuard:* provides an extra layer of safety to the pins. When a pin guard is set, the firmware will automatically set the pin to a defined state after the 'timeout' is reached[7]. This class is not implemented in the simulator since this functionality is relative to a hardware malfunction
- *PinGuardPin*: assigns a pin to the 'pinGuard' class
- *Config:* contains the configuration parameters as firmware parameters (e.g. reply timeouts, etc.), parameters for the movement, etc.
- *Parameter List:* configures the configuration parameters. Default parameters are loaded in case they have not been defined from the user
- *EEProm:* manages the EEPROM memory of the ATMEGA 2560. In our simulator, this class was replaced with a "csv" file used to store the values of the configuration parameters. These parameters are retained even if the Farmbot Simulator is stopped
- *Movement:* responsible for the robot movement and for the update of the robot position
- *ServoControl:* controls the FarmBot servomotors. This class in not implemented since the FarmBot Uniandes works with stepper motors

[7] https://software.farm.bot/docs/pin-guard

- *MovementEncoder:* creates, configures, and obtains the feedback of the encoders. In our simulation, we did not implement this class since the virtual environment does not model physical no-idealities; e.g. obstacles, axis misalignment, etc. For each motor, the actual position has been calculated from the steps commanded to the motor and not from the feedback of encoders
- *MovementAxis:* controls the axis movement using the feedback of encoders and limit-switch sensors. Since encoders were not virtualized in the FarmBot Simulator and the FarmBot Uniandes does not have limit-switch sensors, the actual position has been assumed as the actual position
- *Command:* receives the F- and Q-code functions and converts them into executable commands
- *GCodeProcessor:* instantiates the command to be executed using the functions of the 'GCode Handler' or 'Function Handler'
- *FunctionHandler:* contains all the 'F-code' functions
- *GCodeHandler:* contains all the 'G-code' functions
- *StatusList:* contains the list of possible states; e.g. moving, idle, emergency stop, etc.
- *CurrentState:* sets the current state

Fig. 4. Software architecture of the Farmduino code



The Farmduino Simulator was implemented in Visual Studio and a *graphical display* was generated as shown in Figure 5. In the left-hand side, three FarmBot actuators are placed: the '*Vacuum Pump*' used for the seeding operation, the '*Led Strip*' used to light up the scaled farm at night, and the '*Valve*' used for the watering operation. When one of the actuators is active, its symbol assumes a bright color.

In the right-hand side of Figure 5, the *serial communication* can be configured, initialized and its state is visualized. A '*Command History*' window presents the last commands sent to the simulator.

Eventually, the *FarmBot 'farm'* is represented in the center part of Figure 5. Two orthogonal views are shown enabling the user to locate the FarmBot end-effector in the three spatial dimensions. The movements commanded from the 'FarmBot control software' are scaled to effectively mimic the displacement that the end-effector would have in the physical system. Next, we illustrate how the scaling operation was implemented.

Visual Studio is a *dots per inch* (DPI) aware application, which means that its display is automatically scaled. The scaling operation is performed through a scale factor that can be set in Visual Studio. For example, a 256x256 pixel window will have a 128x128 pixel size if the scale factor is set to 200%. In our case study, we consider a scale factor of 100%. The equivalence in between the '*FarmBot farm´* physical dimensions and the display in pixels is shown in Table 1.

Table 1. Equivalence in between the physical dimensions and the display in pixels

Axis	Physical Dimensions (mm)	Virtual Dimensions (pixels)
X	0-3000	0-360
Y	0-3000	14-366
Z	0-500	17-45

Starting from the table, we were able to identify the equations for converting the physical displacements into the virtual ones:

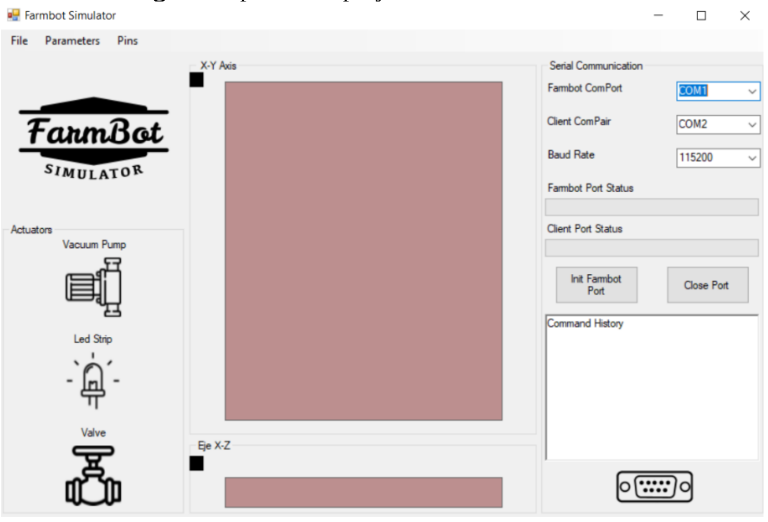
$$d_{px}^{x} = d_{mm}^{x} * \frac{360}{3000} \tag{1}$$

$$d_{px}^{y} = 14 + d_{mm}^{y} * \frac{366-14}{3000} \tag{2}$$

$$d_{px}^{z} = 17 + d_{mm}^{y} * \frac{45-17}{500} \tag{3}$$

Where d_{mm}^{i} is the physical displacement along the i-th axis, and d_{px}^{i} is the virtual displacement along the i-th axis. The physical displacements were calculated from the steps commanded to the motors using the *motor constant* [step/mm] defined within the motor configuration parameters.

Fig. 5 Graphical display of the Farmbot Simulator



Farmbot ComPort COM1
Client ComPair COM2
Baud Rate 115200

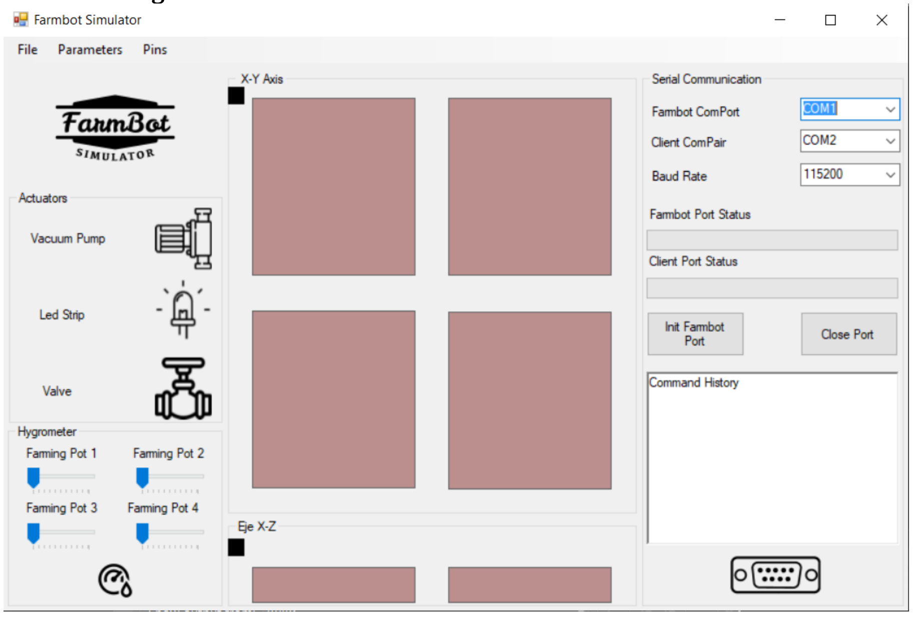
4 Case study

In this section, we use the illustrated SiL simulation for designing a *control software* able to concurrently implement different farming strategies in the FarmBot Uniandes. The variable investigated in the planned experiment is the effect of *soil moisture* in the growth of salad. In the salad cultivar, soil moisture is suggested to constantly remain in between 60-80%[8]. A *hygrometer sensor* is placed on each pot, and irrigation is triggered at different thresholds. Irrigation starts when soil moisture is below the lower bound and stops when its value is above the upper bound. Next, the thresholds planned for each pot are listed:

- Pot 1: $60\% \leq Relative\ Humidity\ (RH) \leq 65\%$
- Pot 2: $65\% \leq RH \leq 70\%$
- Pot 3: $70\% \leq RH \leq 75\%$
- Pot 4: $75\% \leq RH \leq 80\%$

To adapt the *FarmBot Simulator* to the planned experiment, four *pots* are added to the display of the scaled farm as shown in Figure 6. Moreover, four *linear 'Gauges'* are inserted to simulate the soil moisture sensed by each hygrometer. The user can change their values to verify the correct functioning of the system.

Fig. 6 Farmbot Simulator customized for the FarmBot Uniandes



Farmbot ComPort: COM1
Client ComPair: COM2
Baud Rate: 115200

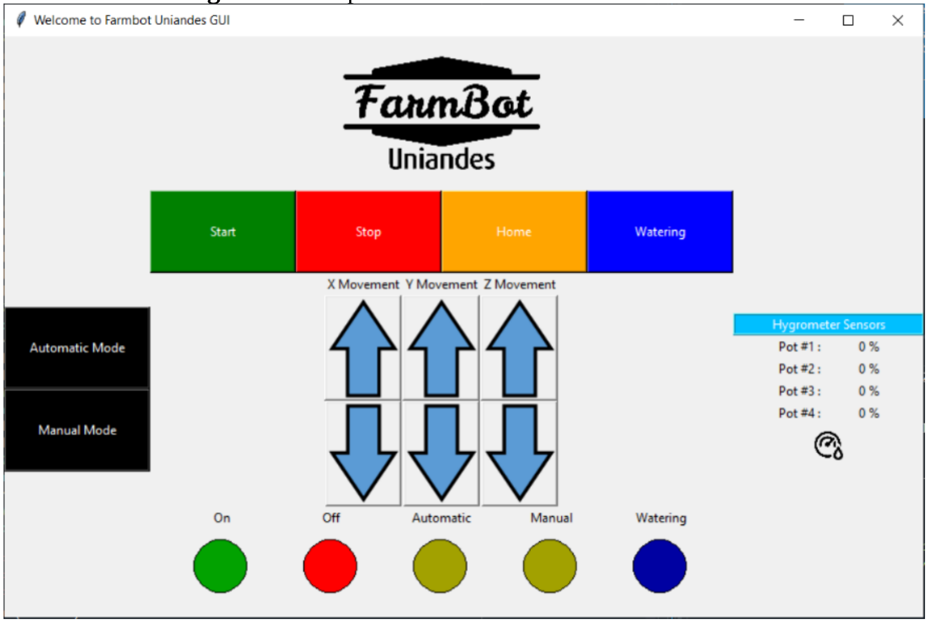
Then, the Python control software is designed using the *Spyder IDE*. Production systems are characterized with different *operational modes*. We decide to implement the following operational modes for the FarmBot Uniandes:

8 https://www.infoagro.com/hortalizas/lechuga.htm

- *Manual operational mode:* the operator can individually move and activate all the FarmBot actuators. This operational mode is useful in case of malfunctions to identify the source of errors
- *Automatic operational mode:* the system implements the desired functionality autonomously, without the need of an operator.

The *HMI* (Human Machine Interface) designed for the FarmBot Uniandes is shown in Figure 7. The HMI is characterized by: (i) *Push buttons*: for controlling the system and switching from one operational mode to the other; (ii) *Leds*: for indicating the status of the system; (iii) *Textboxes*: for showing the values sensed by the hygrometer sensors.

Fig. 7. HMI implemented for the FarmBot Uniandes



Pot #1 :	0 %
Pot #2 :	0 %
Pot #3 :	0 %
Pot #4 :	0 %

5 Results and discussion

Spyder and Visual Studio are interfaced through a virtual serial port using the Free Virtual Serial Ports software. Then, a *SiL simulation* is implemented for the debug of the control software. A video[9] is made available to the reader showing that the developed control software enables the implementation of different PA strategies.

Eventually, we demonstrate how the proposed approach fulfills the *requirements* identified in Section 2. The requirement is recalled on the left-hand side, while the strategy for its fulfillment is placed on the right-hand side:

- *Open-source software* → the SiL simulation is implemented with open-source software complying with the mission of the 'Farmbot project'

[9] https://www.dropbox.com/sh/awc85y637sa1ppm/AAA17VyBk5Lb1YbmrMg_bQ4la?dl=0

- *Interface* → the Farmbot Simulator implements the same interface of the 'FarmBot Operating System'. Seamless deployment of the control software seems viable and will be verified in future work
- *Movement and irrigation* → these two functionalities are implemented in this first version of the simulator and allow the generation of different irrigation strategies
- *Graphical display* → a graphical display able to mimic the system configuration is available to facilitate the debug operation.

In summary, the proposed approach fulfills the objective to verify the control software for the implementation of different PA strategies. Since the FarmBot Simulator consists of a virtual model, it must be clarified that only software errors can be debugged and not errors due to hardware; e.g. not communicating devices, etc.

6 Conclusions and future work

In the context of precision agriculture, this paper proposes a *FarmBot Simulator* for designing PA strategies before its implementation in the physical test-bench. The simulator runs with *open-source software* in accordance with the 'FarmBot project' mission, and implements the same *interface* of the FarmBot controller to enable the seamless deployment of the verified control software.

A case study was developed for demonstrating that the FarmBot Simulator allows the verification of the control software through the implementation of a *Software-in-the-Loop Simulation*.

The developed FarmBot Simulator constitutes a preliminary concept that in the future should be further validated and improved. Some future works identified are:

- *Farming functionality*: in this first version of the simulator only the movement and irrigation functionalities are implemented. A future version will include additional functionalities useful for the development of PA strategies such as seeding, and crop monitoring through the FarmBot Webcam. Moreover, Spyder will be interfaced with the 'Farm Design' web application for the generation of control software that allows to remotely monitor and control the robot
- *Seamless deployment*: the Farmbot Simulator implements the same interface of the 'FarmBot Operating System'. The developed control software will be deployed into the FarmBot Uniandes controller to evaluate the seamless deployment of the verified code
- *PA test-bench*: a farming experiment will be implemented to certify that the FarmBot can be utilized as a test-bench for PA.

References

1. Fukuyama, M. (2018). Society 5.0: Aiming for a new human-centered society. Japan Spotlight, 1, 47-50.
2. United Nations (2014). World urbanization prospects: The 2014 revision-highlights. Department of Economic and Social Affairs. Retrieved from: https://esa.un.org/unpd/wup/publications/files/wup2014-highlights.pdf

3. World Bank Group. (2016). Global Monitoring Report 2015/2016: Development Goals in an Era of Demographic Change. Retrieved from: http://pubdocs.worldbank.org/en/503001444058224597/Global-Monitoring-Report-2015.pdf
4. Tilman, D., Cassman, K. G., Matson, P. A., Naylor, R., & Polasky, S. (2002). Agricultural sustainability and intensive production practices. Nature, 418(6898), 671-677.
5. Kumar, R. R., & Cho, J. Y. (2014). Reuse of hydroponic waste solution. Environmental Science and Pollution Research, 21(16), 9569-9577.
6. Mitsch, W. J., Gosselink, J. G., Zhang, L., & Anderson, C. J. (2009). Wetland ecosystems. John Wiley & Sons.
7. McBratney, A., Whelan, B., Ancev, T., & Bouma, J. (2005). Future directions of precision agriculture. Precision agriculture, 6(1), 7-23.
8. Zapata, F., Barbieri, G., Ardila, Y., Akle, V., & Osma, J. (2019) AgroLab: a living lab in Colombia for research and education in urban agriculture. Cumulus Conference Proceedings - The Design After.
9. Aronson, R. (2019). FarmBot. Retrieved from: https://farm.bot/
10. Lee, C. G., & Park, S. C. (2014). Survey on the virtual commissioning of manufacturing systems. Journal of Computational Design and Engineering, 1(3), 213-222.
11. Maranville, S. (1992). Entrepreneurship in the business curriculum. Journal of Education for Business, 68(1), 27-31.
12. Schilling, M. A., & Shankar, R. (2019). Strategic management of technological innovation. McGraw-Hill Education.
13. Reinhart, G., & Wünsch, G. (2007). Economic application of virtual commissioning to mechatronic production systems. Production engineering, 1(4), 371-379.
14. Lechler, T., Fischer, E., Metzner, M., Mayr, A., & Franke, J. (2019). Virtual Commissioning–Scientific review and exploratory use cases in advanced production systems. In 26th CIRP Design Conference (Vol. 81, pp. 1125-1130).
15. Koo, L. J., Park, C. M., Lee, C. H., Park, S., & Wang, G. N. (2011). Simulation framework for the verification of PLC programs in automobile industries. International Journal of Production Research, 49(16), 4925-4943.