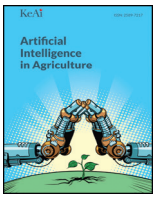
Artificial Intelligence in Agriculture 4 (2020) 172–183

Contents lists available at ScienceDirect

Artificial Intelligence in Agriculture

journal homepage: http://www.keaipublishing.com/en/journals/artificial-intelligence-in-agriculture/



Design of a 4 DOF parallel robot arm and the firmware implementation on embedded system to transplant pot seedlings

Rahul K., Hifjur Raheman *, Vikas Paradkar

Agricultural & Food Engineering Department, Indian Institute of Technology Kharagpur, India

ARTICLE INFO

Article history:
Received 17 July 2020
Received in revised form 14 September 2020
Accepted 14 September 2020
Available online 21 September 2020

Keywords:
Robot firmware
Parallel robot arm
Automation
SoC microcontrollers, pot seedlings

ABSTRACT

This paper presents a firmware design and its implementation on a real time embedded system for driving a 4 DOF parallel robot arm. The firmware primarily comprised of two components to produce motion of the robot arm: a) generation of continuous position coordinates and b) generation of actuating signals. These two components were processed in two different microcontrollers with a common communication bus. The position generation algorithm produced and transmitted continuous position data to the motion generation algorithm in the form of G-code strings by reading the input positions which were previously stored by the user in EEPROM memory of the microcontroller. The receipt of a handshake signal synchronized the data transmission between these components through a communication bus. An LCD display and keypad were used as human-machine interface (HMI) to communicate with the user to set the robot target coordinates. The mechanical structure of the robot arm comprised of multiple links which were actuated by stepper motors. The workspace boundary were sensed by limit switches. The kinematic equations represented the gripper position for the corresponding input joint angles. A microcontroller was used to compute kinematic equations of the robot arm with the help of motion generation algorithm to generate actuation signals for the simultaneous movement of robot joints. The kinematic equations were solved with the dual-core capability of the microcontroller using real time operating system (RTOS), which made the computation faster with an average computation time of 198 μs per step. The developed firmware was implemented and tested on a 4 DOF parallel manipulator using embedded microcontrollers for continuous pickup and place of the paper pot seedlings for automating the metering of pot seedlings. The cycle time taken for pickup and dropping of each seedling was 3.5 s with a success rate of 93.3%.

1. Introduction

A robotic system generally consists of a motion generating mechanical hardware module, an electronic control module, a software module that run a decision support system to drive the electromechanical actuators and a communication protocol for data transfer within and outside the system (Angeles, 1997). The electronic module would consist of embedded microcontrollers, PLC's, or other advanced hardware such as FPGA's to generate the actuating signals. These signals are then received by the motor drivers to supply driving power to the actuators. Power regulators are used to provide different operating power levels for each component according to its working range. Also, multiple sensors are used to monitor the environment (Joseph Raj et al., 2019) and the robot operating parameters (position measuring devices such as encoders, limit switches and potentiometers) (Chen et al., 2018). The electromechanical actuators such as stepper and servo motors provide driving torque to the robot joints through power transmission mechanisms like gears, timing belts and pulleys with minimal backlash. The robot firmware implemented on PC or embedded microcontroller plays a crucial role in developing the necessary actuation signals by making decisions through task allocation, algorithm computation and data transmission. Among the various modules of the robotic system, the firmware implemented on the embedded microcontroller decides the control and action of each module to perform the predefined tasks. The typical function of a robot firmware is shown in Fig. 1.

The firmware can fetch and modify data from storage memory to perform the tasks according to the instructions provided by the user. It drives the actuators by reading the sensors connected with the robotic system. It decides to channel the data transfer either internally or externally to read, write or display data through the required data bus.

1.1. Literature review

Robot operating system (ROS) is an open-source firmware architecture, mainly designed for the development of robotic tools. It consists of several libraries which are capable of performing complex tasks such as navigation robots, multiple robot coordination assisted with machine vision and sensor integration. Araujo et al. (2015) developed a mobile robot platform using an Arduino board controlled by ROS. The control

* Corresponding author.
E-mail address: hifjur@agfe.iitkgp.ac.in (H. Raheman).

https://doi.org/10.1016/j.aiia.2020.09.003

Nomenclature

θ_a	Active arm joint angle
θ_p	Passive arm joint angle
θ_{tr}	Transmission angle
α	Passive arm angle with respect to active-passive joint orientation
β	Active arm linear angle with horizontal
γ	Inner angle of active arm
C(x,y)	End effector position in Cartesian plane
COM bus	Communication bus
EEPROM	Electrically Erasable Programmable Read Only Memory
HMI	Human Machine Interface
k and s	Linear distance between origin of active arms and end effector position
L_1	Length of active arm
L_2	Length of passive arm
LCD	Liquid crystal display
m	Horizontal distance between the active-passive joints
M_{dep}	Elbow position memory address for dropping position
M_{dp}	Dropping point memory address
M_{pep}	Elbow position memory address for pickup point
M_{pp}	Memory address of pickup position value
n	Vertical distance between the active passive joints
O	Origin of the robot axis
PGF	Position Generation Firmware
SGF	Signal Generation firmware

logic was developed in a PC and the command signals were sent to Arduino through a USB communication. This robot was built for educational purposes to demonstrate swarm robotics, multi-robot surveillance and rescue tasks. The robot used an onboard camera to navigate and send the environment data wirelessly. Kamel et al. (2017) developed a model in ROS for trajectory tracking technique for unmanned aerial vehicles (UAV). Though ROS firmware architecture is designed for solving complicated robotic operations, advanced hardware with high computation power is required to deploy the code blocks developed in ROS firmware. Therefore, still there is a requirement for a simple and efficient firmware structure that could be implemented on a commercially available system on chip (SoC) microcontrollers.

The most notable open-source firmware available for driving a robotic system using SoC is 'Marlin' firmware (Wijnen et al., 2016; Kruger et al., 2018). Marlin is an open-source software constructed using C++ language, dedicatedly developed for operating 3D printers using embedded microcontrollers. Some typical functions of Marlin firmware are: a) positioning the printing nozzle in x, y and z coordinates according to the commanded locations by a slicing software, b) driving the extruder for continuous feeding of the printing filament, c) heating the filament and printing bed by PID control with the feedback of temperature sensors. Marlin takes the input of NC (G-codes and M-codes) codes (Arroyo et al., 2004) to operate the 3D printer, which produces real 3D components by melting an input filament at high temperature through an extruder. A list of essential G-codes for positioning the printing nozzle of a 3D printer is given in Table 1.

Tiansong et al. (2019) developed a 3D printed 3 DOF serial robot using open source Marlin firmware controlled by ATMEGA 1284P microcontroller. The robot was actuated by geometric codes generated by Pronterface software installed on a desktop personal computer (Surange and Gharat, 2016). Nejatimoharrami et al. (2017) developed a liquid handling robot equipped with machine vision for automating chemical handling. It is a gantry type robot controlled by Marlin firmware installed on an Arduino Mega microcontroller board. Although Marlin architecture could be modified for operating a robotic system using commercial microcontrollers, it has several drawbacks such as:

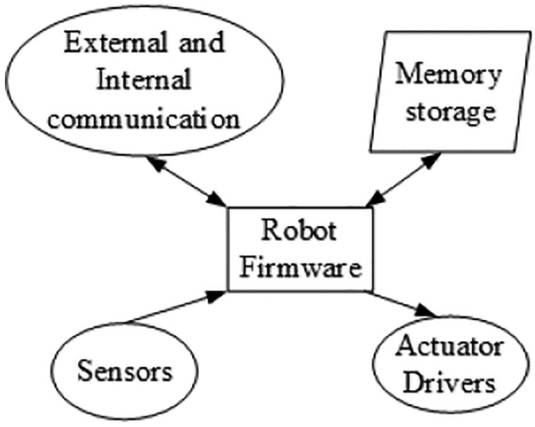


Fig. 1. Functions of a robot firmware.

Marlin software requires an additional system to input the positional coordinates. Generally, the robots run by Marlin firmware uses an external PC to generate the positional G-code strings. Also, the amount of memory requirement of Marlin firmware is high because of additional modules written for the functioning of 3D printer components. The operation speed cannot be improved significantly due to its core design (3D printers does not require high operation speed) and it requires an external code generation module to direct the actuators.

Many a researchers have been using Marlin firmware to develop a robotic system to perform several useful tasks. This external software supplies the necessary codes to manipulate the robotic system. It creates difficulty in operating the robotic system in a hostile environment and with space constraints. This paper presents a firmware architecture and implementation algorithms to run a 4 DOF parallel robot, which could be deployed in an embedded system that comprised of SoC microcontrollers, HMI and electronic components which eliminated the need of an external PC to generate the positional coordinates. To implement and validate the developed firmware, it was implemented on a real time embedded hardware, a robot arm based vegetable transplanter was selected to meter the seedlings of vegetables raised in paper pots.

1.2. Automation of vegetable transplantation

Vegetable transplanters are used in agriculture to simplify the manual process of planting already grown seedlings in agricultural field to improve the yield and reduce drudgery of the human labor (Kumar and Raheman, 2011; Dihingia et al., 2018). Pot seedling transplanters generally consist of a seedlings pickup mechanism (extracts seedlings from a feeding conveyor and place it into a drop tube one by one), conveying unit (to continuously transfer seedlings towards the pickup unit) and a soil opening-closing unit. Researchers have been working on automating this mechanical transplanting process by including robots as a picking mechanism to improve its efficiency and to reduce the input power requirement (Hwang and Sistler, 1986; Hu et al., 2014; Rahul et al., 2019). The robot used in these transplanters picked up seedlings one by one from multiple numbers of fixed pickup locations and dropped it into a fixed dropping point (Fig. 2).

To maintain a delicate picking without damaging the seedlings and to drop the picked seedlings without collisions with other components

Table 1
List of essential G-codes for the nozzle spatial manipulation of 3D printer.

Code	Function
G00 X Y Z	Rapid linear movement
G01 X Y Z	Linear motion of nozzle at a specified speed
G28	Homing of X, Y and Z coordinates
G02 and G03	CW and CCW arc at a specified radius

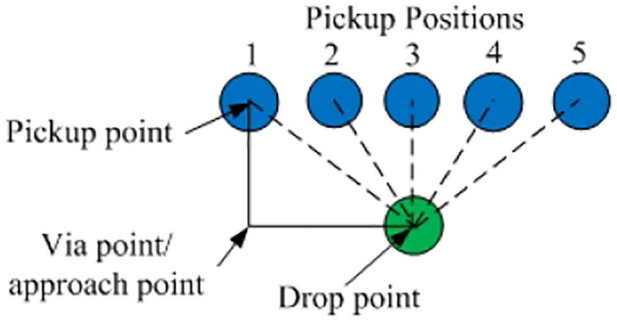


Fig. 2. Robot pickup and drop positions with a via point trajectory.

of the transplanter, the manipulator needs to move from dropping point (initial coordinates) to each of the pickup locations through a via point. After picking, the manipulator again reaches the drop point through the same via point. The via point is known as the approach point for each pickup location. A systematic motion requirement of a robot for seedling handling was reported by Rahul et al. (2019). The optimal robotic path requires an approach point, gripping point and a dropping point. The operational speed of picking should be lesser than the speed of approach and dropping. This entire problem for the robotic task is considered as a pick and place application with multiple fixed pickup positions and one drop point at variable operating speed. Hence, the hardware and software embedded control system for the robot in an automated transplanter requires the following features:

1) A user interface system to modify the robot motion coordinates and its parameters without the aid of an external computer
2) A fault diagnosis system to detect the status of sensors and actuators whenever required
3) Display of the operating parameters (i.e., number of droppings and execution code received) through HMI

In this study, a robot control firmware was constructed and deployed on a real time embedded system to control a 4 DOF parallel manipulator robotic system used for continuous pick and place operations of paper pot seedlings in an agricultural field. The details of its developments and its laboratory evaluation are given in the following sections.

2. Materials and methods

2.1. Firmware architecture of the robotic system

The proposed firmware architecture of the robot system is shown in Fig. 3. It consisted of two components: 1) signal generation firmware (SGF) and 2) position generation firmware (PGF).

A user interactive HMI was used to store or modify the data for pick and place task such as the number of positions, approach point, pickup point and dropping point in the EEPROM memory at their specified memory addresses. Also, an external communication bus was used to monitor and control the robotic system by a remote computer. During the pick and place operation, PGF read the position data (X, Y, Z location) from EEPROM memory and compiled it into a G-code string (Ex. G01 X d_1, Yd_2, Zd_3; where d_k is the k^{th} position data). The generated G-code was then sent to the SGF and waited for the handshake signal to indicate the completion of the assigned task. The SGF read the input string and interpreted the X, Y and Z position coordinates. The coordinates were used to compute the trajectory points by Bresenham 3D line algorithm. For each trajectory point, the algorithm calculated the inverse kinematics of the robotic system (in this case, a planar parallel robot was selected) to compute the joint angular positions at that instance. The required joint motor actuation signals were generated and supplied to follow the

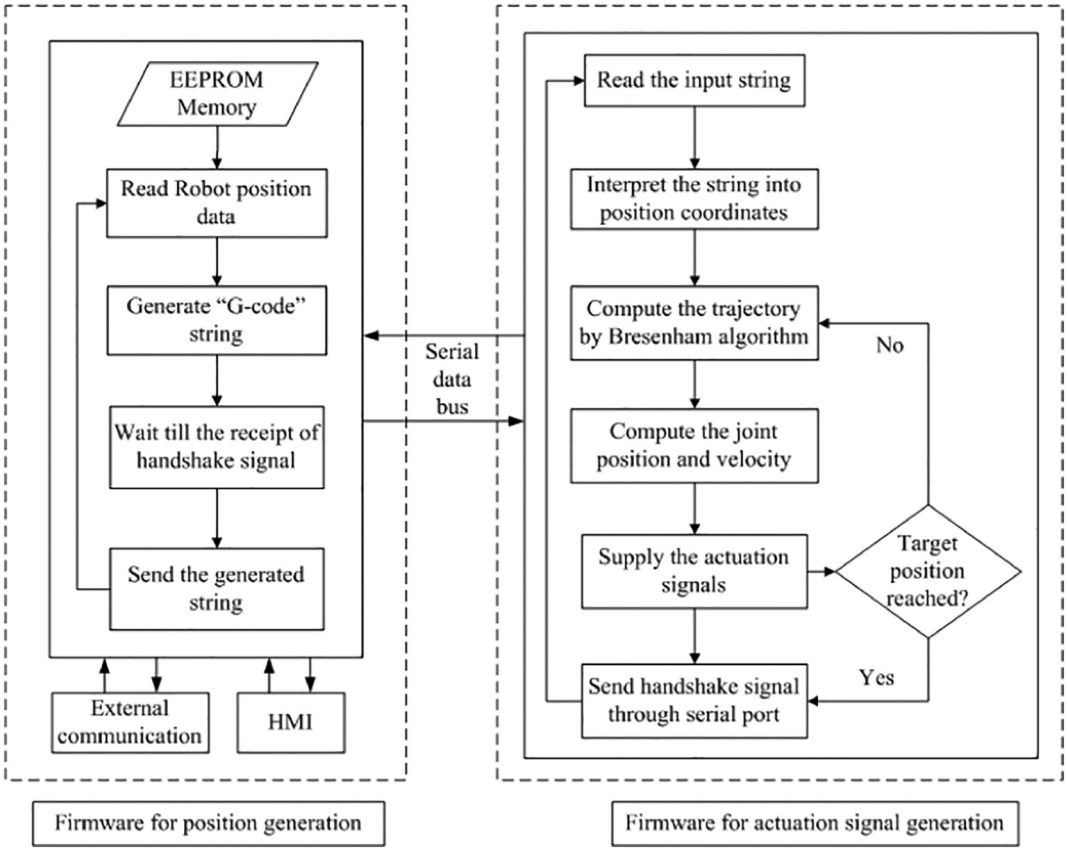


Fig. 3. Generalized firmware architecture of the robotic system.

trajectory at the specified linear speed of the manipulator. Further, acceleration at the beginning and deceleration at the end of the trajectory were provided to maintain smooth actuation. Once, the manipulator reached the target position, SGF sent a handshake signal to the PGF through serial bus to continue the cycle for the next position.

2.2. Design of robot arm

The validation of the proposed firmware architecture for operating a robotic system was made by a) designing the mechanical structure of the robot arm, b) deriving the kinematic equations of the robot arm, c) designing the embedded hardware to actuate the mechanical system and d) implementing the firmware in a deployable SoC microcontroller of the hardware. The following sections explain the practical implementation of the proposed firmware architecture.

2.2.1. Design of mechanical structure of robot arm

The 3D CAD model of a 4 DOF parallel robot arm is presented in Fig. 4. It consisted of two active links those were driven by stepper motors through timing belt and pulley. One end of the two passive links was joined together and the other ends were connected with the corresponding active links. Thus the combined motion of the active joints provided a planner manipulation in X and Y direction.

The arm assembly platform, which comprised the robot links and actuation stepper motors was mounted on the support frames through a slider with a linear bearing that permitted motion only in 'Z' direction. Another stepper motor mounted on top of the frame drove a lead screw coupled with the platform. Thus, the rotation of the stepper motor was converted into a linear motion in Z direction. A parallel jaw gripper was attached to the robot arm at the connecting point of the two passive links.

The gripper was made to open and close using a servo motor. Another servo motor mounted on the passive link was attached to the gripper to provide the rotary motion of the gripper in planner coordinates. Four limit switches were provided at the robot arm frames to act as sensing boundary of the robot arm. Two limit switches (limit switches 1 and 2) sensed the boundary position of the active robot joints. The limit switches 3 and 4 detected the upper and lower boundary positions of the platform, in Z-up and Z-down directions, respectively.

2.2.2. Inverse kinematics of the robot arm

The input joint angles θ_{a1} and θ_{a2} for the given gripper position were computed with inverse kinematics. The geometric line diagram of the parallel manipulator is shown in Fig. 5 with three operating regions to visualize easily and to develop the implementable robot equations. The regions are separated based on the position of the end effector, i.e., C(x, y) in the planar space.

Using cosine rule in triangle O_1AC (referring Fig. 5), the inner angle of active arm 1.

$$\gamma_1 = \operatorname{acos}\left(\frac{{L_1}^2 - {L_2}^2 + k^2}{2\,L_1\,k}\right) \text{ where, } k = \sqrt{(x + L_0)^2 + y^2} \quad (1)$$

Similarly, using cosine rule at triangle O_2DC, the inner angle of active arm 2

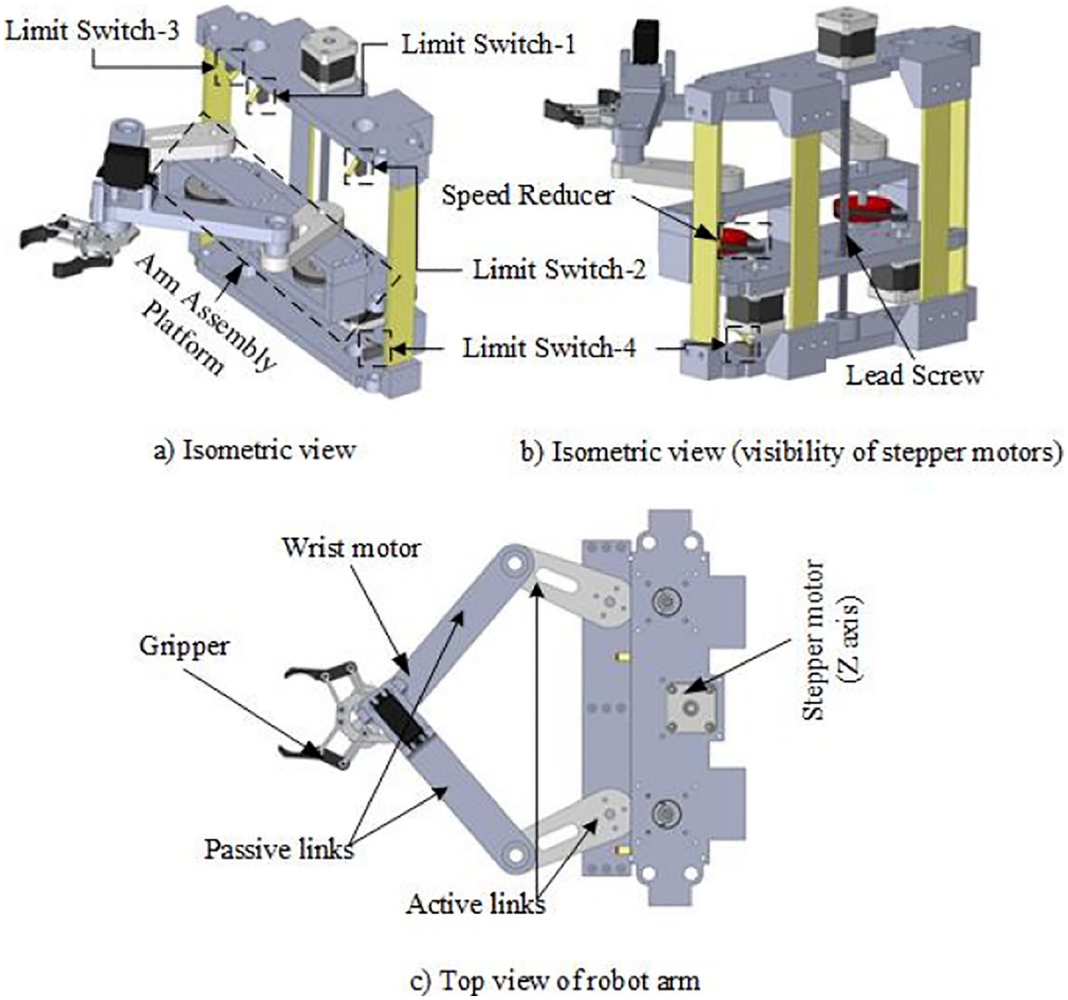


Fig. 4. CAD model of the developed robot arm to implement the proposed firmware.

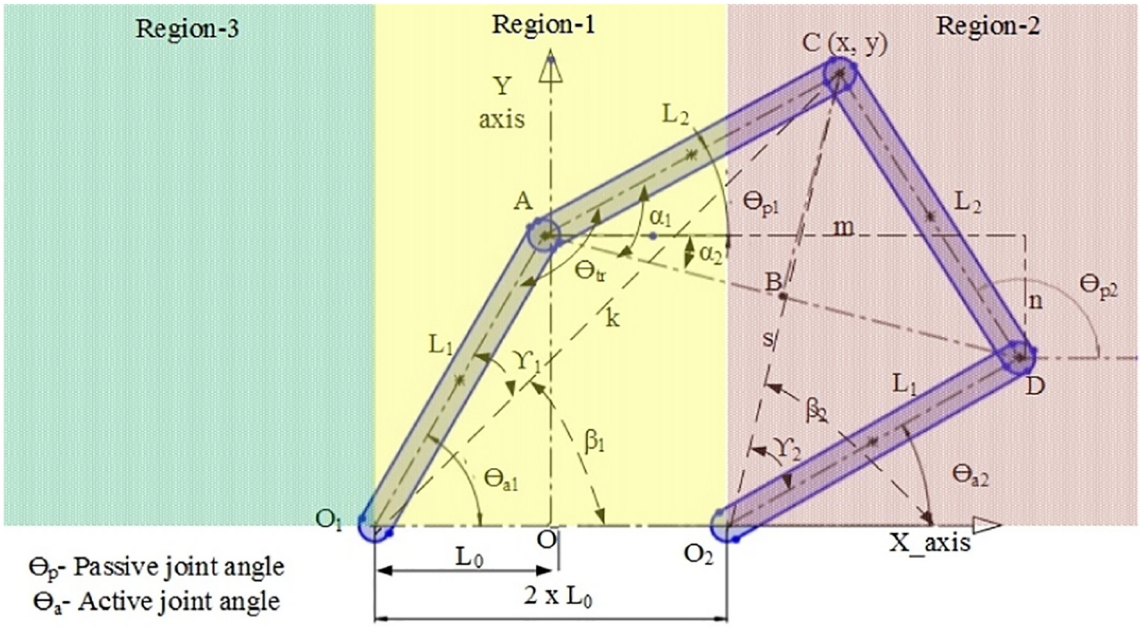


Fig. 5. Geometric line diagram of the manipulator with its operating regions.

$$\gamma_2 = \text{acos}\left(\frac{{L_1}^2 - {L_2}^2 + s^2}{2\,L_1\,s}\right) \text{ where, } s = \sqrt{(x - L_0)^2 + y^2} \tag{2}$$

Based on the location of the manipulator point 'C' within the region, the active arm linear angle with horizontal (β_1 and β_2) varies as:

$$\begin{aligned}\beta_1 &= \text{atan}\left(\frac{y}{|L_0 + x|}\right) \text{ and } \beta_2 \\ &= \text{atan}\left(\frac{y}{|L_0 - x|}\right) \text{ for the location of 'C' in region} - 2\end{aligned} \tag{3}$$

$$\begin{aligned}\beta_1 &= \text{atan}\left(\frac{y}{|L_0 + x|}\right) \text{ and } \beta_2 \\ &= \pi - \text{atan}\left(\frac{y}{|L_0 - x|}\right) \text{ for the location of 'C' in region} - 1\end{aligned} \tag{4}$$

$$\begin{aligned}\beta_1 &= \pi - \text{atan}\left(\frac{y}{|L_0 + x|}\right) \text{ and } \beta_2 \\ &= \pi - \text{atan}\left(\frac{y}{|L_0 - x|}\right) \text{ for the location of 'C' in region} - 3\end{aligned} \tag{5}$$

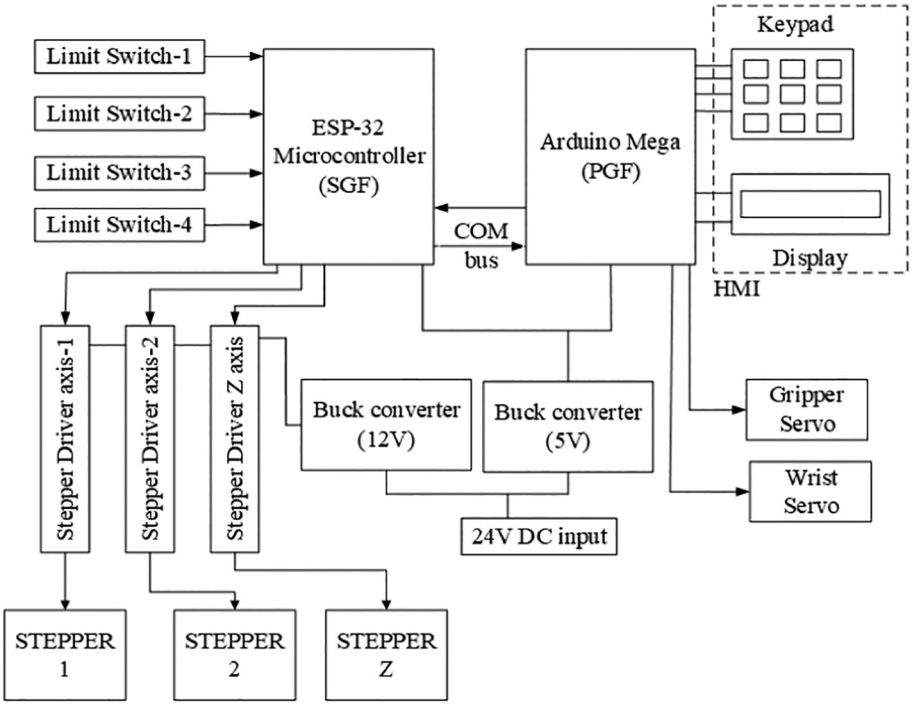


Fig. 6. Block diagram of the electronic control circuit of the robot arm.

Table 2
List of components used in electronic hardware.

Component	Specifications	Functions
ESP 32	Clock frequency: 240 MHz CPU: 2 cores (32 bit), Tensilica LX6 microprocessors Memory: 520 Kib SRAM Peripheral interfaces: 3 UART, 2 I2C	SGF firmware installed to drive the robot joint actuators
Arduino Mega	Clock frequency: 16 MHz CPU: 8 bit ATMEGA 2560 Memory: 8Kb SRAM Peripheral interfaces: 4 UART, 1 I2C	PGF firmware installed to generate and supply the motion control codes
Buck converter	Input: 36 V DC (max), current: 5A (max) Output: 5 V and 12 V	To supply recommended constant DC power to actuators and microcontrollers
HMI	Keypad LCD	To receive input from the user and to display the operating parameters
Actuators	NEMA 17 Stepper motors and Servo motors (up to 180° rotation)	To provide driving torque to the robot joints

The active joint angles (θ_{a1} and θ_{a1}) are determined by

$$\theta_{a1} = \beta_1 + \gamma_1 \tag{6}$$

$$\theta_{a2} = \beta_2 - \gamma_2 \tag{7}$$

The transmission angle influences the manipulator stability θ_{tr} between active and passive links and is directly proportional to the distance between points B and C; which is determined by using the following equations.

$$\text{The vertical distance of link, } L_{1y} = L_1 \sin\theta_{a1} \tag{8}$$

$$\text{The vertical distance of link, } L_{2y} = L_2 \sin\theta_{a2} \tag{9}$$

$$\text{Length between points B and C, } L_{BC} = \min\left(L_{1y}, L_{2y}\right) + \frac{abs\left(L_{1y}, L_{2y}\right)}{2} \tag{10}$$

2.3. Design of embedded hardware for the robot arm

The electronic system incorporates the necessary hardware components to realize the robot arm motion. The block diagram of electronic components used and their associated connections are shown in Fig. 6. The list of hardware components used with their specifications and functions are summarized in Table 2.

The SGF firmware deployed on dual-core ESP 32 microcontroller actuated the stepper motors of robot joint by supplying the essential signals to the corresponding motor drivers. It calculated both robot kinematic and trajectory equations. Limit switches were also connected to the ESP 32, to sense the robot joints whenever it reached the robot boundary. The PGF firmware installed on the Arduino Mega board took input from the HMI keypad and displayed the necessary data through an LCD. The Mega board also controlled the gripper open/close and wrist rotation of the robot arm. A COM bus connected these two microcontrollers, transferred data between them. Buck converters provided the required DC voltage to these electronic components for their operation and actuation.

2.4. Robot path planning

Robot path planning involves the generation of optimal reachable points in spatial coordinates from the present position of the manipulator to the target location. Command G00 (rapid motion) and G01 (motion of

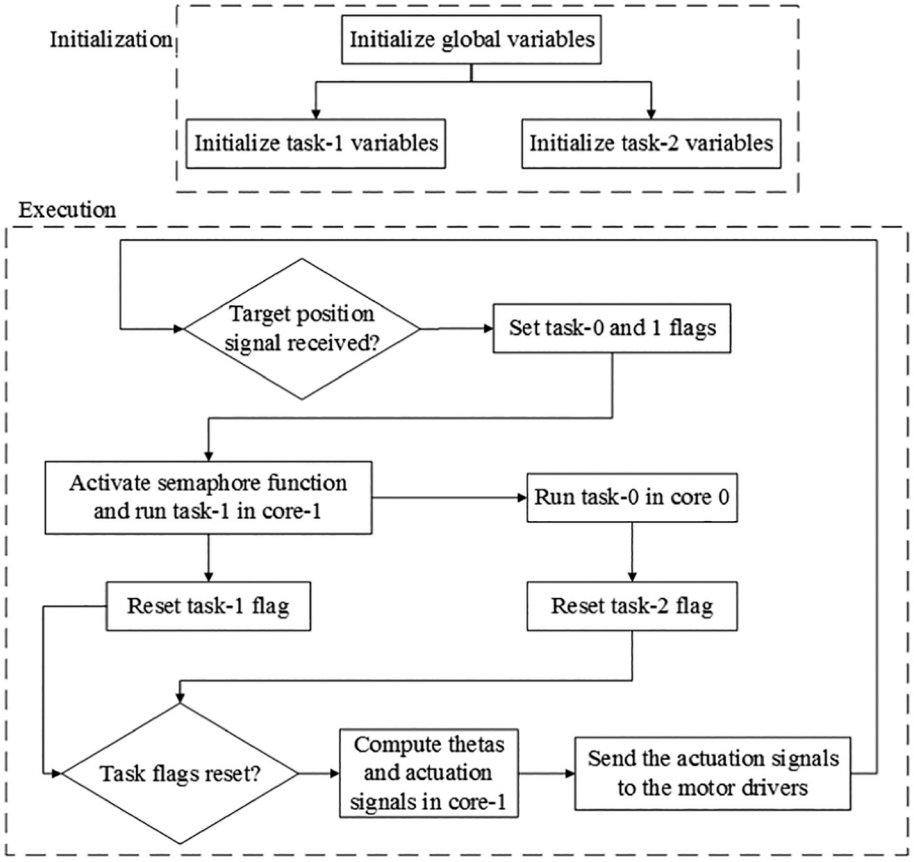


Fig. 7. Flow diagram of robot kinematics computation through dual-core.

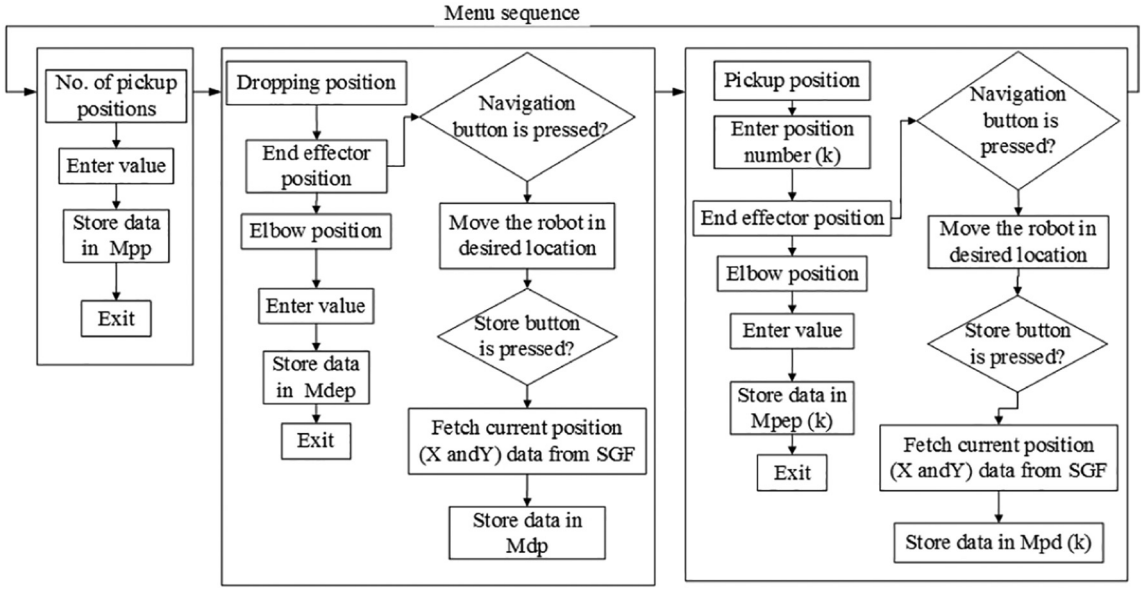


Fig. 8. User interactive menu sequence to store data in EEPROM memory.

manipulator at specified speed) were used to move the end effector between the specified coordinates (X, Y and Z) with a straight-line trajectory using Bresenham algorithm.

2.5. Computation of kinematic equations through dual-core

The robot kinematics involves the computation of complex geometric equations to determine the angular positions and velocities of the joints at minimal time. As discussed in the earlier section, higher the resolution, better is the straight-line approximation, but at the cost of increased computation. Hence, the processor needs to calculate the kinematic equations at a higher speed to supply the actuation signals to the corresponding stepper motors. By utilizing the advantage of dual-core computing capability of ESP 32 microcontroller, the complex algorithms were solved at a minimal time. Since all the kinematic equations were required to be solved only by the SoC microcontroller, the equations were divided into two parts, each was computed individually in different cores and the results were

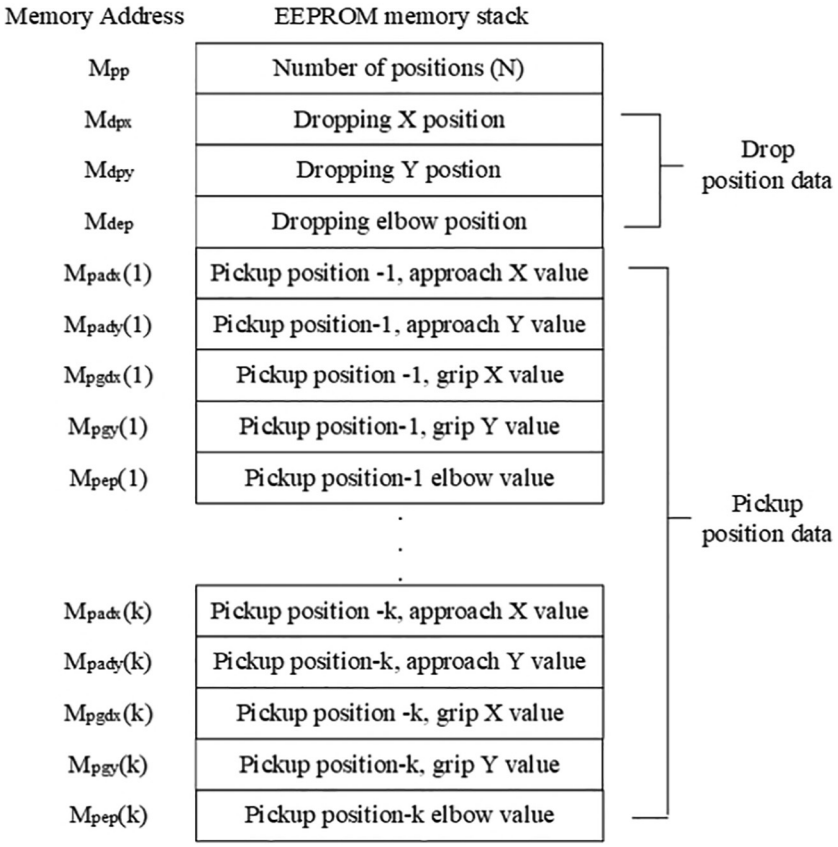


Fig. 9. EEPROM memory stack and allocation of data in its associated storage locations.

combined to obtain the target angular position. The algorithm for enabling dual-core computation is given in Algorithm 1 and the block diagram of dual-core computation is shown in Fig. 7. ESP32 was supported with RTOS (Real Time Operating System); thus multiple tasks were created and ran those tasks simultaneously in different CPU cores. Two sets of kinematic equations ran independently in cores 1 and 2 by using 'Semaphore' (give or take) function by defining global variables that shared common data between both the cores. Finally, the computed data from both the cores were collected and the target angular positions θ_{a1} and θ_{a2} were calculated.

Algorithm 1. Dual-core computation of kinematic equations.

Start

Initialize the global variables, L0, L1, L2, X, Y and Z

Initialize the tasks:

Task 1:

s, γ_1(eq. 1)$and\ \beta_1(eq.3\ or\ 4\ or\ 5, depend\ on\ the\ region\ of\ manupulator)$

Task 2:

$k, \gamma_2\ (eq.2)\ and\ \beta_2\ (eq.3\ or\ 4\ or\ 5, depend\ on\ the\ region\ of\ manupulator)$

If target coordinates are received from Bresenham line algorithm (X, Y and Z)

Set task flags in core 0 and core 1

If task flag is set

Begin Semaphore function to enable second core computation

Run the dual core (core1 and core0) computing using global variables

Reset the corresponding flags after the completion of tasks

end If

Compute θ_{a1}, θ_{a2} and actuation steps for steppers in core 1 by the processed data

Supply the actuation signals to the stepper motors to reach the target position

end if

end

2.6. Construction of position generation firmware

The position generation firmware produced the robot control codes (G-codes) required for the actuation of the robot arm to perform a particular task (i.e., in this case, continuous pick and place application of pot seedlings was considered). The generated signals were sent through the internal serial communication bus (COM bus) of the robot controller upon the receipt of handshake signals from the receiver. In order to produce the continuous control codes which perform pick and place task, the controller was required to access the position data. Considering the case of pick and place application as explained in the Introduction section, the user could provide all the required position data at once which would be stored in EEPROM memory. Then the controller could access the data at any time repeatedly without consideration of power supply ON/OFF criteria. Besides, provision was also made to change specific data in the memory according to the user's requirement.

The position data storage for the robot actuation was made through a user interactive menu-based system as shown in Fig. 8. Navigation buttons of the keypad were used to change the menu sequence to access a particular address of the EEPROM memory (Fig. 9). The number of pickup positions value was stored in memory address M_{pp} by navigating the corresponding menu and by entering a numeric entity. The dropping position data were stored in the memory by directing to 'End effector position' in dropping position menu. With the help of a keypad, the end effector could be moved to the desired position and the corresponding coordinates (X and Y) could be fetched from the SGF through COM bus to store the dropping point data in memory address M_{dp}. Elbow position for the dropping coordinate were stored in address M_{dep}. In this particular

Start

Initialize variables:

Z axis lift value (Z_{up}), Z axis down value (Z_{low}),

Lower speed value (S_l), Higher speed (S_h), initial current position (C = 1)

Fetch the following position data from EEPROM:

M_{pp}, M_{pdx}, M_{pdy}, M_{dep}, M_{padx}[C], M_{pady}[C], M_{pgdx}[C], M_{Pgdy}(C) and M_{pep}(C)

Generate and send string for approach for position 'C',

G01 X(M_{padx}[c]) Y(M_{pady}[c]) Z (Z_{low}) S (S_h)

Wait for acknowledgement and go to next line after its receipt

Generate and send string for gripping position 'C',

G01 X (M_{pgdx}[c]) Y(M_{pgdy}[c]) Z (Z_{low}) S (S_l)

Position elbow at M_{pep}[c] and Open gripper

Wait for acknowledgement and go to next line after its receipt

Generate and send string for gripping position 'C',

Close gripper

G01 X (M_{pgdx}[c]) Y(M_{pgdy}[c]) Z (Z_{up}) S (S_l)

Wait for acknowledgement and go to next line after its receipt

Generate and send string for move back to approach locatition 'C'

G01 X(M_{padx}[c]) Y(M_{pady}[c]) Z (Z_{up}) S (S_h)

Wait for acknowledgement and go to next line after its receipt

Generate and send string for move to dropping location

G01 X (M_{dpx}[c]) Y(M_{dpy}[c]) Z (Z_{up}) S (S_h)

Position elbow at M_{pep}[c] and Open gripper

Increment 'C' by 1

If 'C' > M_{pp} then reset value of 'C' to 1

If Menu button is pressed, go to menu sequence

Go to Fetch position data from EEPROM for the updated 'C'

End

application, there were multiple numbers of pickup points; which required multiple end-effector coordinates and elbow positions. These data were accessed and modified by the user by entering the pickup position number 'k' in menu sequence which could access the memory locations $M_{pdx}(k)$, $M_{pdy}(k)$ and $M_{pep}(k)$ and these particular data were stored in the specific memory address.

Algorithm 2. Continuous pick and place operation.

The procedural steps for the continuous pick and place operation are provided in Algorithm 2. It initialized the required variables and the data for pick and place were fetched from the EEPROM memory (Fig. 9). It generated the G code string required for the robot motion for approach coordinates and transmitted the string to the SGF which actuated the elbow and gripper. After the receipt of acknowledgement signal from SGF, the algorithm generated strings for gripping and drop coordinates for the pick and place of the present position. It incremented the position data and actuated the robot until it was equal to M_{pp}. The position data was again reset back to pick and place from the initial point.

3. Results and discussion

3.1. Data entry through HMI

The HMI system was connected to the PGF microcontroller to display data and to provide input to the SGF microcontroller for the actuation of the robot arm. The developed HMI control for the robotic system, which comprises an LCD display and Keypad is shown in Fig. 10. The buttons of the keypad were configured for multiple functions (up, down, left, right, select, back and numerical entries) to access the data.

The entry of position data in EEPROM memory of the PGF microcontroller is shown in Fig. 11. By navigating the menu system through the up/down and select buttons user could obtain 'Enter value' tab in the display (Fig. 11.a). User could change the number of pickup positions by the buttons (Fig. 11.b). Then by pressing the select button, the entered data were saved in the corresponding EEPROM memory location of the microcontroller (Fig. 11.c). This updated data were used by the PGF for the next operation cycle.

The data entry for the dropping position in the corresponding memory address of the EEPROM is shown in Fig. 12.

3.2. Operation of robot arm by the proposed firmware implemented on the embedded microcontrollers

The developed firmware (both SGF and PGF) were deployed in SoC microcontrollers of the embedded hardware and it was implemented on a 4 DOF parallel robot (Fig. 13).

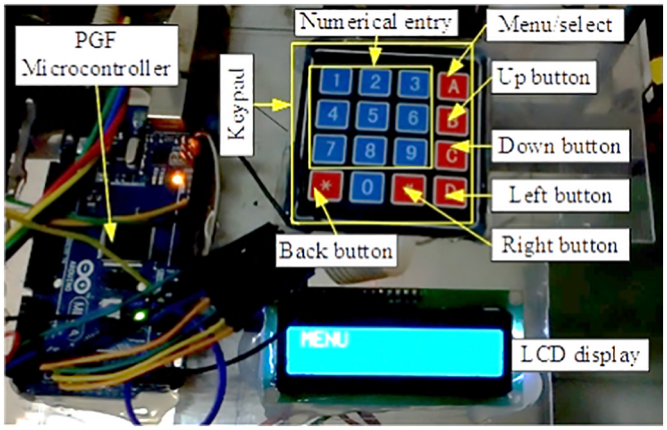


Fig. 10. HMI system for the robot arm.

It consisted of five pickup positions and one dropping position. The algorithm was tested for continuous pick and drop of pot seedlings loaded on a feeding conveyor. Forty numbers of pot seedlings were picked and dropped by rotating the feeding conveyor to supply pot seedlings to the pickup positions. The average computation time (for 20 pickup positions i.e., four row of pot seedlings) required to solve the kinematic equations were recorded. Function 'micros ( )' was used in the kinematics code block within the SoC microcontrollers of three different modes and the data were recorded. Table 3 presents the comparison of the average computation time to solve the computation of inverse kinematic equation in different modes.

It was found that, ESP 32 SoC microcontroller computed the kinematics at less time as compared to the utilization of widely used Arduino board. Further, usage of dual-core capability through RTOS improved the computation speed, which increased the operating speed.

3.3. Performance evaluation of the developed firmware implemented on the hardware to pick and drop the pot seedlings

The robot arm was tested for seedling pickup and dropping. A total of 40 pot seedlings were loaded on the cells of the feeding conveyor for testing. The sequence of operation for picking and dropping of seedlings by the robotic arm is shown in Fig. 14(a–d).

The cycle time taken for pickup and dropping of each seedling was 3.5 s (which maintained delicate picking and dropping of pot seedlings without any damage at an average of 17 seedlings per min which is sufficient for walk behind type vegetable transplanters, Kumar and Raheman, 2011). The average success rate ($S = \frac{\text{Number of successfully picked seedlings}}{Totalnumberofseedlingsloaded} \times 100$) of the robot arm deployed with the proposed firmware for three trials was computed as 93.3%.

3.4. Comparison of the proposed firmware and the associated hardware with existing systems

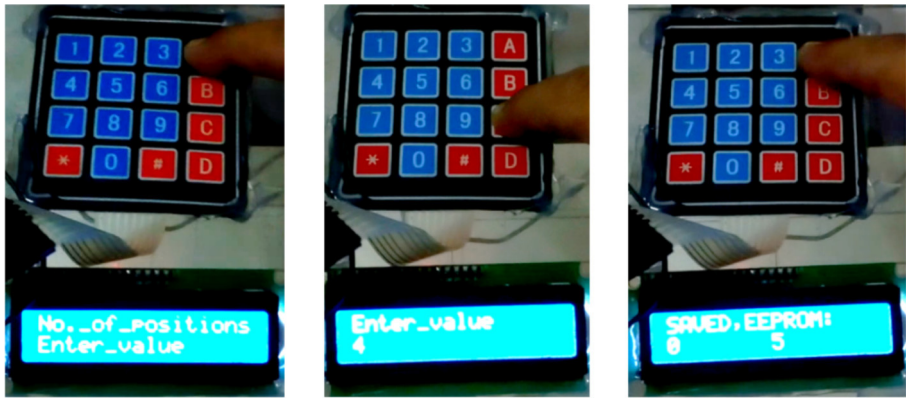
It is required to compare the developed firmware and the hardware on which the firmware was deployed. Table 4 presents the comparison of various features of the developed firmware and the associated hardware with the existing systems.

The SMC4-4-16A 16B CNC motion controller has built in firmware and already well established in several industrial NC machines. In cost wise this controller is not suitable for operating with agricultural machines. Compared to Raspberry pi based ROS system, the developed firmware with its associated hardware is much lower in cost. Also, the performance of the developed system proved that it is well suited for metering pot seedling for transplanting.

4. Conclusions and future work

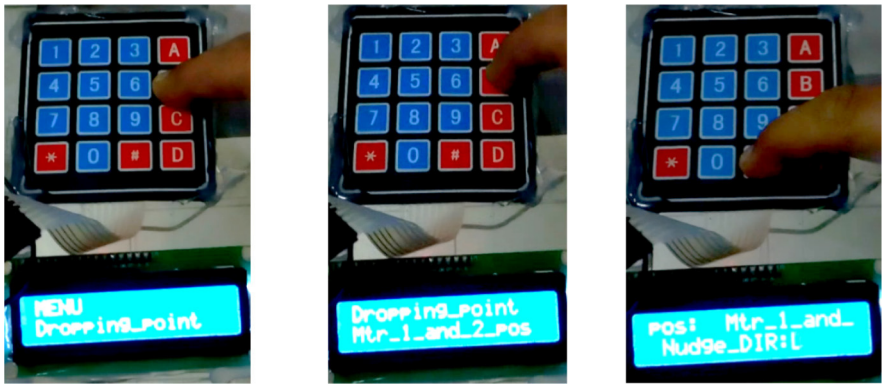
In this paper, design of a 4 DOF parallel robot arm, the development of control firmware and the implementation of the developed firmware on embedded system to handle paper pot seedlings for vegetable transplantation were studied. A general design of a robot firmware was proposed to produce control codes (G-codes) based on the predefined user input and motor actuation signals obtained for the internal commands received, and by sensing the joint angular positions. An embedded electronic hardware was developed and deployed the proposed firmware to generate control codes and actuation signals using SoC microcontrollers (tested using ESP 32, Arduino Mega microcontrollers, LCD-Keypad HMI, limit switch position sensors and the required power converters). 3D Bresenham algorithm was used in the firmware to approximate the linear trajectory of the parallel robot in between initial and target positions.

Region-based inverse kinematic equations were developed and implemented in the robot firmware, coupled with the Bresenham line algorithm to realize the linear trajectory of the robot in real time. The



a) Navigate to positions b) Numerical entry c) Saving the data

Fig. 11. Modifying position data in memory through HMI.



a) Navigate to drop menu b) Motor positions selection c) Move the robot joint

Fig. 12. Setting of motor positions for drop point in memory through HMI.

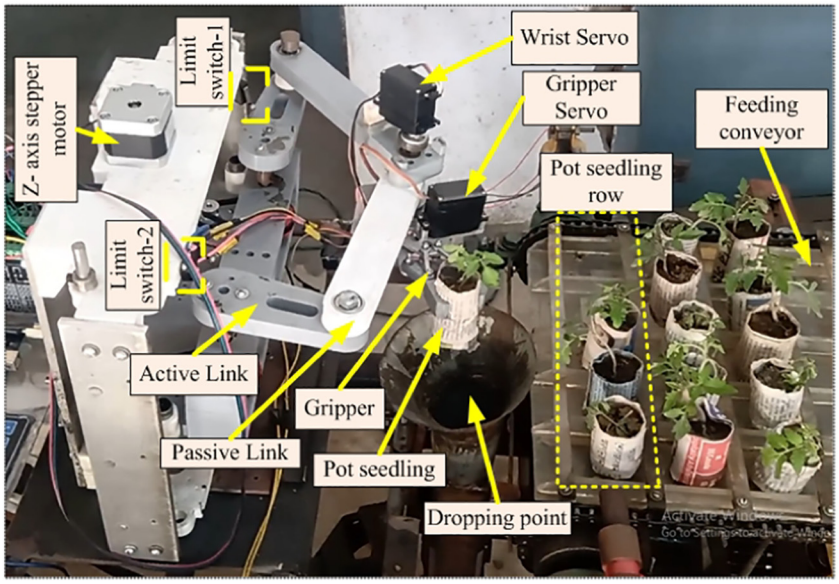


Fig. 13. Robot arm actuated by the proposed firmware for pick and drop of pot seedlings.

Table 3
Time taken to solve the kinematic equation with different microcontrollers.

Device used	Average computation time per step (μs) (For 20 different positions)
Arduino Mega	4027
ESP 32 (without dual-core)	276
ESP 32 (dual core enabled)	198

kinematic equations were computed at a lesser time using the dual-core capability of the ESP 32 microcontroller, which resulted in an average computation time of 198 μs per step as compared to 276 μs and 4027 μs with single-core mode and other commercial microcontrollers. A user interactive menu sequence was developed to store the position data related to pick and drop in the EEPROM memory using HMI. The developed firmware was successfully implemented on an embedded hardware to pick and drop of pot seedlings loaded on a feeding conveyor using a parallel robot arm with an average cycle time of 3.5 s with a success rate of 93.3%.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Angeles, J., 1997. Fundamentals of Robotic Mechanical Systems: Theory, Methods, and Algorithms. 2nd edition. Springer-Verlag, New York.

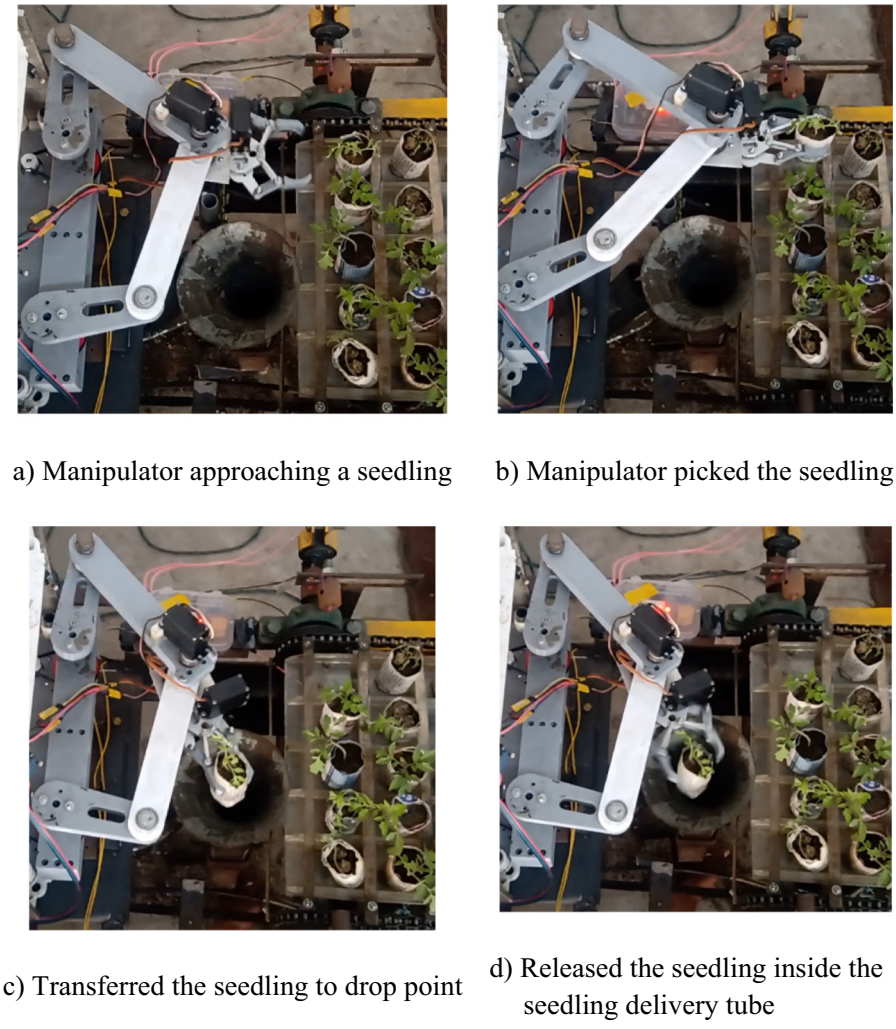


Fig. 14. Pick and drop of a pot seedling with robot arm under laboratory conditions.

Table 4
Comparison of developed firmware and its associated hardware with existing systems.

Robot control firmware with hardware	Cost (approx.)	Number of controllable axis	Applicability
Proposed system (Microcontrollers with HMI)	Rs. 1800	Up to 5 axis	Robotic applications
Raspberry pi with ROS and HMI	Rs. 4000	Up to 6 axis	Robotic applications
SMC4-4-16A 16B	Rs. 60,000	Up to 4 axis	Industrial CNC machines

Araujo, A., Portugal, D., Couceiro, M.S., Rocha, R.P., 2015. Integrating Arduino-based educational mobile robots in ROS. J. Intell. Robot. Syst. 77 (2), 281–298.
Arroyo G, Ochoa C, Silva J and Vidal G., 2004. Towards CNC Programming Using Haskell. In Ibero-American Conference on Artificial Intelligence. Springer, Berlin, Heidelberg, pp. 386–396.
Chen, W., Khamis, H., Birznieks, I., Lepora, N.F., Redmond, S.J., 2018. Tactile sensors for friction estimation and incipient slip detection - toward dexterous robotic manipulation: a review. IEEE Sensors J. 18 (22), 9049–9064.
Dihingia, P.C., Kumar, G.P., Sarma, P.K., Neog, P., 2018. Hand-fed vegetable transplanter for use with a walk-behind-type hand tractor. International Journal of Vegetable Science 24 (3), 254–273.
Hu, J., Yan, X., Ma, J., Qi, C., Francis, K., Mao, H., 2014. Dimensional synthesis and kinematics simulation of a high-speed plug seedling transplanting robot. Comput. Electron. Agric. 107, 64–72.
Hwang, H., Sistler, F.E., 1986. A robotic pepper transplanter. Appl. Eng. Agric. 2 (1), 2–5.
Joseph Raj, A.N., Sundaram, R., Mahesh, V.G., Zhuang, Z., Simeone, A., 2019. A multi-sensor system for silkworm cocoon gender classification via image processing and support vector machine. Sensors. 19 (12), 2656.
Kamel, M., Stastny, T., Alexis, K., Siegwart, R., 2017. Model predictive control for trajectory tracking of unmanned aerial vehicles using robot operating system. Robot Operating System (ROS). Springer, Cham, pp. 3–39.
Kruger, J., Gu, W., Shen, H., Mukelabai, M., Hebig, R., Berger, T., 2018, February. Towards a better understanding of software features and their characteristics: a case study of marlin. Proceedings of the 12th International Workshop on Variability Modelling of Software-Intensive Systems, pp. 105–112.
Kumar, G.P., Raheman, H., 2011. Development of a walk-behind type hand tractor powered vegetable transplanter for paper pot seedlings. Biosyst. Eng. 110 (2), 189–197.
Nejatimoharrami, F., Faina, A., Jovanovic, A., St-Cyr, O., Chignell, M., Stoy, K., 2017. UI Design for an Engineering Process: Programming Experiments on a Liquid Handling Robot. In 2017 First IEEE International Conference on Robotic Computing (IRC). IEEE, pp. 196–203.
Rahul, K., Raheman, H., Paradkar, V., 2019. Design and development of a 5R 2DOF parallel robot arm for handling paper pot seedlings in a vegetable transplanter. Comput. Electron. Agric. 166, 105014.
Surange, V.G., Gharat, P.V., 2016. 3D printing process using fused deposition modelling (FDM). Int. J. Res. Eng. Technol. (IRJET) 3, 1403–1406.
Tiansong, L., Feng, G., Yilong, Y., 2019. Design of low-cost desktop robot based on 3D printing technology and open-source control system. 2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC). IEEE, pp. 739–742.
Wijnen, B., Anzalone, G., Haselhuhn, A., Sanders, P., Pearce, J., 2016. Free and open-source control software for 3-D motion and processing. Journal of Open Research Software 4, e2. https://doi.org/10.5334/jors.78.

Rahul K. is persuing Master of Science at Indian Institute of Technology Kharagpur in Farm Machinery specialization under Agricultural and Food Engineering Department. His research focuses on implementation of robotics and automation in agricultural machinery. He is also interested in artificial intelligence, machine vision and mechatronic system design.

Hifjur Raheman is a Professor in the Agricultural and Food Engineering Department of Indian Institute of Technology Kharagpur, India. His research areas are use of robotics in agri-machinery, renewable energy operated agricultural machinery, alternate fuels from biomass (Biodiesel, producer gas) and bioelectricity production and tillage and traction improvement in tractors and power tillers.

Vikas Paradkar is Ph.D Student, Specialization in Farm machinery and Power, Agricultural and Food Engineering Department, Indian Institute of Technology Kharagpur, India. His research focus is on implementation of robot in vegetable transplanter.