Vol. 06, No. 4 (2024) 1685-1694, doi: 10.24874/PES06.04.027

Proceedings on Engineering Sciences

www.pesjournal.net

DEVELOPMENT OF A MICROCONTROLLER BASED ROBOT AND ITS USE IN TEACHING CONTROL AND NAVIGATION

Samuel Oliver Wane [1]
Matthew Butler
Anthony James Bautista

Received 16.04.2024.
Received in revised form 10.09.2024.
Accepted 27.09.2024.
UDC – 007.52

Keywords:

Mobile Robot, Navigation, Microcontroller, Control, GPS, UTM Coordinates, Teaching Control

A B S T R A C T

This paper sets out a framework for the essential elements necessary for the implementation of outdoor autonomous mobile robots along with example algorithms. Robotics is a multi-disciplinary subject requiring a hands-on approach when developing machines to interact in the real world. Courses in robotics exist which either include simulated robots, access to robots in a laboratory, or remote access to robots in a controlled indoor environment. This study presents a teaching system based on a robust robot with a focus on navigation and communications in an uncontrolled outdoor agricultural environment. An all-terrain robot teaching platform was developed with the necessary hardware, libraries, and teaching material consisting of lectures and practical field-work. The course is tested on a variety of student levels and understanding is assessed with practical demonstration. This article shares the teaching philosophy and an overview of the hardware and software.

1. INTRODUCTION

The self-driving tractor of the Hands Free Hectare project clarified Harper Adams University as a leader in applied autonomy in agricultural vehicles. This then developed into the Hands Free Farm where routes between fields and machinery sheds were autonomously traversed. While these projects inspire, the sheer size and complexity of such systems can put people off. The core algorithms used for the navigation of such vehicles are actually basic and can be demonstrated and programmed by young people with an interest in the subject (figure 1).

To develop on this, a robot teaching platform (Bautista & Wane, 2018) was developed with the necessary hardware and supporting programs. The teaching material which consisted of lectures and practical field-work were designed around the building of self-driving off-road robots which have parallel uses in our research work. What is presented here are a highlight and an implementation of the key algorithm elements necessary for robot programming.

Engineering is the application of technology in the real world, and a huge motivator is when the student puts theory into practise and uses their intuition coupled with knowledge and experience. Problem solving games such

[1] Corresponding author: Samuel Oliver Wane
Email: swane@harper-adams.ac.uk

as 'Bridge Builder' allow the student to develop at their own pace, see the results immediately and gain a 'feel' for the right answers. This intuition step is a sort of self-checking and sanity check on their calculations.
A majority of the applications in control and robotics are to develop systems that act in the real world, and the teaching of control and robotics engineering topics are well received when it is related to the physical environment. Computer code can be brought to life when it interacts with the environment, and this is an infinitely changing testbed for developing code.

An online method for teaching mobile robots was developed where the students can remotely access the robot in a laboratory (Kulich et al., 2013) which unfortunately seems to longer be running.

Robot competitions inspire learning and can encourage schools to adopt classes in robotics. Whilst difficulties are experienced in terms of equipment cost and getting schools involved, many hurdles can be overcome with the use of simulated robots, programming environments were also found to be advantageous. The work in (Shoop & Flot, 2016) reports on Robot Virtual Worlds developed at Carnegie Mellon University, which was developed and sponsored as part of the National Science Foundation NSF DRK-12 (Barker, 2012). Specialist competitions can focus on specific subject applications, for example, the Field Robot Event is hosted each year in Europe and focusses on robots acting autonomously in a farm field environment.

The use of real or simulated robots encourages a constructivism approach where knowledge is gained through experience, and with the right teaching materials, a mental model of abstract concepts is followed by practical experience and can then be formalised through theory. The programming concepts are not necessarily robot specific but are good programming solutions put into practise. A 'bookend approach' defines a particular problem, allows students to think of possible solutions, followed with practical work and experience, and then backed up with reflection on what worked and if the targets were met.

Unique to our study is the agricultural applications of robotics, we have given students practical experience in getting robots outside and testing them in the nearby fields, allowing them to realise the application of algorithms on an all-terrain vehicle, and this is also a unique selling point for our robotic courses. Harper Adams University specialises in agriculture and has a strong ethic of putting theory into practise. However, the need is greater than teaching programming concepts. The global citizen challenge is focussed on testing and implementing concepts and strategies to engage young people in efforts that will reduce inequality related to the first six Global Goals of: No poverty, Zero hunger, Good health and well-being, Quality education, Gender equality, and Clean water and sanitation. We believe that robots, particularly applied to agriculture, target at least three of these.

There are organisations and voluntary groups which attempt to address these issues on a global scale through the use of robotics. In "We Robotics" they teach local people to be able to apply drones to aid humanitarian efforts, applying technology in real problem areas, and educating local people in the latest technological advances at the same time. There is a global interest in autonomous vehicles, and it is not just cars. Autonomous tractors are being developed worldwide by major tractor manufacturers such as John Deere, Fendt, Case IH, and the Autonomous Tractor Corporation.

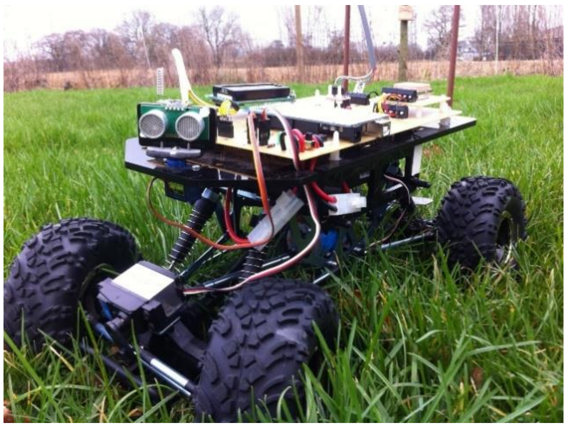
Figure 1. The Atlas Vehicle

The role of robotics in agriculture is the next essential step and a natural progression of agricultural development stemming from the replacement of the stick with the plough, the oxen with the tractor, and the hoe with the spray. The case for small mobile robots working in fleets in fields rather than a single large tractor can reduce overall compaction, they can wait for the optimal moment when the wind drops when spraying, and can be used in autonomous weeding and pest control (Blackmore et al., 2010). Selective spraying and harvesting and the reduction on the reliance of imported labour has made the use of robotics in agriculture an essential option. The rise of the population requiring middle-class diets, and the decreasing land mass has alerted the governments in many countries to the need for automation.

There are not enough engineers skilled in the robotics and agricultural technology areas to push the industry forward into a smart, efficient, sustainable future. The younger generation are being lured into urban areas as these are perceived to be technology rich with a high demand for computer based work. However, there are many facets of robots spanning the food chain and there is huge development in the areas of robotic farming. We need to promote robots as safe, reliable machines, which can respond to dynamic situations in order to shift the attitudes of the public towards them.

The goal is to teach microcontroller interfacing and programming, navigation using a GPS receiver and a compass, robot behaviour models, feedback and control, communication, and mapping. The results of the robot navigating a set course will inform us of how well these have been achieved.

2. METHODOLOGY

We required a robot, small enough to rest on a desk and to be physically carried to a field, with an accessible and available programming environment and community, ability to cope with rough terrain, front-wheel steered (as found on a majority of mobile farm machinery) and contain its own power source.

- Microcontroller:-The microcontroller chosen was the Arduino Mega, as it has a huge online community backing and is extremely versatile with libraries (pre-written code) in C++ developed for many items of robot sensors. The programming language is freely available and students are able to install the Arduino Integrated Development Environment (IDE) on their own Personal Computer. Moreover, it has been adopted for teaching microcontrollers in our university engineering department.
- Chassis:-The chassis used was the Rock Crawler type of radio controlled car, a four-wheel drive radio controlled all-terrain vehicle. It has a front and rear servo steer motors and a DC motor for front and rear drive and comes with batteries and a charger. Figure. 1 shows the completed system in-situ. The radio receiver unit was replaced with our custom designed PCB (figure 2) consisting of a variety of sensors necessary for autonomous control in an outdoor environment. A library was written to enable easy use of the sensors and motors allowing the students to focus on the application of the peripherals without the problems of wiring and demystifying sensor communication protocols.
- Communication:-The serial wireless device HC-11 is used as a simple RS232 wireless transceiver. The range is approximately 80m and each transmission can be received by many receivers-which can allow for broadcast and networks of communication. A Bluetooth transceiver is also used for communication with a mobile App.
- Compass: A compass is required to determine heading and used for steering control to a particular waypoint. This measures the direction of magnetic flux and has to be mounted away from the motors and chassis. It has a resolution of 0.5º.
- Accelerometer: The accelerations in the x,y,z directions is available from the open source library 'FreeSixIMU'. This data is mostly used by the students to determine the tilt of the robot by detecting the components of gravity in each axis, but can teach the use of integration to infer velocity and position.
- This creates simple keywords to perform sensor calibration and unit conversion value from sensors (e.g. read distance in cm), control motors to angles or velocities (e.g. steer 40º), and to build more complex behaviours (e.g. follow wall, drive to GPS point). Table 1 summarises the sensors, interface and some library commands to access the sensor data.
- GPS:-A GPS receiver gives the ability to determine latitude and longitude (to an accuracy approximately of ±3m), altitude, degrees of precision, number of satellites, time, and date. The latitude and longitude are useful for the display of the robot's position using Google Maps, but for autonomous navigation they are converted into Northing and Easting values in the UTM (Universal Transverse Mercator) coordinate frame using a custom written function inspired from the IBM repository.
- Range sensor:-An ultrasonic sensor works on time of flight of a sound pulse and is connected to a pan motor – allowing it to sweep an area and allows the programming of basic obstacle avoidance (for behaviour programming), to determine the best direction around an obstacle, and for wall following (when panned to the left or right). The sensor along with pan motor emulates the expensive and large laser distance scanners commonly used in robot navigation today.
- LCD: A basic two-line monochrome display works well in the outdoors and can display text and variables. Data is sent using the RS232 protocol although students access it using a simple 'print' command.

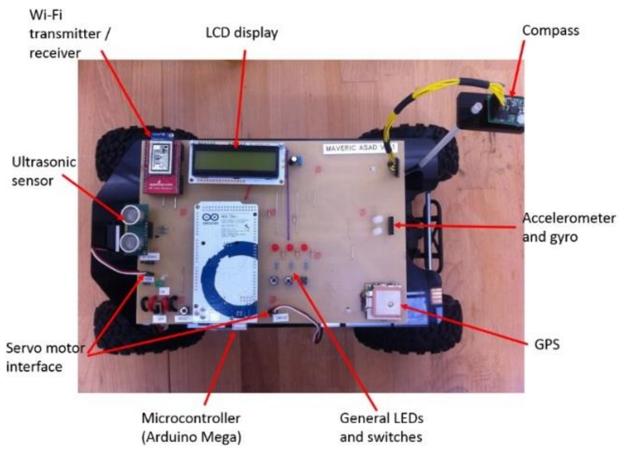
Wi-Fi transmitter / receiver
LCD display
Compass
Ultrasonic sensor
Accelerometer and gyro
Servo motor interface
GPS
Microcontroller (Arduino Mega)
General LEDs and switches

Figure 2. The layout of the PCB

Table 1. The sensors and actuators interface

Sensor	Connection	Details	Function
GPS	RS232	Reports an NMEA string embedding time, date, latitude, longitude etc	get_lat, get_lon
Compass	I2C	Reports orientation in degrees	read_compass
Accelerometer and gyroscope	I2C	Angular and velocity and linear acceleration in three axes	get_roll, get_pitch
Ultrasonic sensor	Pulse width	Range 2-80cm	read_ultrasonic
Switches	Digital	3 x input switches for program control	read_button(1)
LCD	RS232	Two line includes Clear screen and Next line control codes	clear_LCD() display(text)
LEDs	Digital	Direct digital connection	led_out(1,ON)
Steer motor	Digital PWM	Controls steer angle	steer(10)
Drive motor	Digital PWM	Uses a motor driver to control DC motor	drive(1)

- Steer: The steering motor was calibrated in the library so the programmer addresses it in degrees, with 0^o –straight ahead, $+40^o$-maximum right and -40^o–maximum left. This matches with the compass orientation of clockwise positive. The drive motors are DC motors with a L298 motor driver that uses PWM and is calibrated to accept speed up to ±1 m.s^{-1} (negative for reverse).

The general layout of the PCB mounted on the chassis is shown in Figure. 2. Some components have been removed for ease of viewing, the ultrasonic sensor is usually on a rotating platform.

3. LESSONS TYPES

Learning the application of control methods

For these classes, Control is divided into logical control, where when an input reaches a threshold-it sets an output, and continuous control-where the real-world sensed values are used to proportionally control an output.

Logic parameters of AND, OR and NOT are introduced along with the 'if' and 'else' statement by reading a button and controlling the LEDs: If the antecedent tested through the logic function to the consequent is true, then run the section of code associated with it:

If (THIS IS TRUE) SECTION OF CODE TO RUN

Lessons can teach the student to program the system to light LEDs or other outputs when combinations of inputs are within certain ranges using the less-than and greater-than in the 'if' statement. The Truth Table is introduced and logical deduction is practiced.

Edge detection , debounce, and rate of change of input

The importance of the changing input is introduced with examples which include how to count a person entering a shop-breaking a beam ensuring only a single count, how to compensate for the mechanical bounce in switch contacts, and how to determine the rate of change of input (velocity) with time and the extra useful information this can give, for example by not only knowing the distance to an object but if the object is quickly approaching or not as in the equation:

$$velocity = \frac{\Delta input}{\Delta time} \quad (1)$$

The command 'velocity_input', takes any variable and, storing the previous value, divides the change in value by the time between function calls.

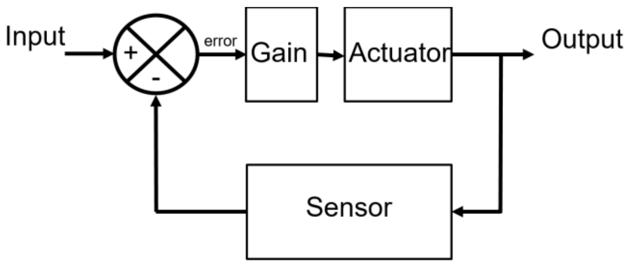

Figure 3. The closed-loop feedback diagram

Closed Loop Control

Proportional feedback is the principal foundation of Control Theory, it is introduced practically using the closed loop diagram and applied in a practical programming environment. The Input is a desired value, this could be the desired bearing the robot is to aim for, desired lateral distance to a wall it is following, desired angle of trailer to the robot body when reversing (e.g. 0^o when reversing in a straight line), desired distance to a point, etc. The students are shown the block diagram and then how computer code relates to this Figure. 3, and a gain element demonstrated to alter the sensitivity.

The gain element is the crucial factor here, it can simply be a number greater or less than one, but it can also be a function, e.g. a Sigmoid function taking the error value (difference between the desired and actual value) and outputting a value depending on the error. For example, the function in Figure. 4 results in mild steering changes for small errors and more severe values as the error increases.

$$y = 10log_{10}\left(\frac{x}{1-x}\right) \quad (2)$$

The equation (2) and shown in Figure 4 Can be realised in computer code as:

```
y=10*(log10(x/(1-x)));
```

Where y is the output steer angle and x is the input error (where error=desired distance – sensed distance)

The standard closed-loop diagram can explain this item of program code clearly. The system doesn't have to be calibrated (i.e. the steering angle is not calculated), the feedback is constantly correcting the actuator to achieve the desired value, and this sensitivity can be altered with a gain which can by tuned using observed response. A summary of the Control applications is given in Table. 2.

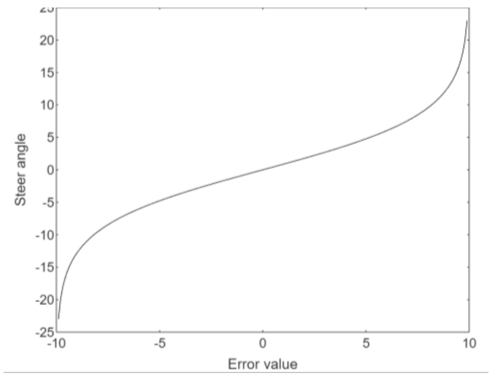

Figure 4. A non-linear control gain

Value approximation, calibration, filtering

Analogue values will always have noise, students quickly learn that to switch on an exact value (e.g. when the distance equals exactly 0.29m) rarely works. Values must have a threshold boundary to fall within so:

$$Des - th \leq input \leq Des + th$$

Where: Des is the desired value, th is the threshold, and $input$ is the input value.

The result is TRUE when the input value is within the range of $Des \pm th$. The function below realises this:

```
approx(input,Des b,th)
```

Table 2. Input-output linked to a behaviour

Application	Input	Output
Wall follow	Distance to wall	Steer angle
Steer to a bearing	Actual bearing	Steer angle
Maintain velocity	Wheel speed	Drive power
Follow light	Angle of bright light	Steer angle
Maintain straight line	Angular velocity	Steer angle

There are calibration functions to match the compass bearing to latitude/longitude coordinate bearing and steer angles, with the offsets and min/max values being stored in EEPROM.

$$b = b_0 + (b_1 - b_0) \times \frac{a-a_0}{a_1-a_0} \quad (3)$$

Using equation (3) maps the original values a_0 and a_1 (bit values) into the physical units b_0 and b_1 (millimetres, temperature, etc):

b =mapf(a, a_0, a_1, b_0, b_1)

The output from any analogue sensor can be graphed on-screen, and the need for filtering can be seen. A digital first order filter of the equation:

$$y(k) = f_c Tx(k-1) + y(k-1) - f_c Ty(k-1) \quad (4)$$

Where f_c is the cutoff frequency in $rad.sec^{-1}$, T is the sample time in seconds, x is the input, k is the current sample, and y is the output.

Steering to a bearing

A way of explanation to make the robot always steer in a particular direction, and this can be tested practically.

There are certain practical issues when using Control Theory (actuator limits, methods of driving, non-linear sensors, etc.) and these are described as follows: Referring to Figure. 5, a desired bearing (Input) of say, South (180^o), is given, the actual bearing (Output) is read (Sensor), and the difference (error) is multiplied by a small gain (Gain) and changes the steering angle (Actuator) within the actuator limits! A desired bearing (Input) of North (0^o) presents is own problems, the non-linearity rollover of 359^o-0^o is discovered and the need for a solution is demonstrated with the library function 'closest_bearing_difference' to resolve this to a difference of ±180^o, i.e. the difference between a heading of 350^o and 15^o should be 25^o rather than 335^o, and between a heading of 20^o and 300^o should be -80^o rather than 280^o.

The gain section can be a number, a PID gain (acting on the dynamics of the system), or a non-linear value (exponential or parabolic function based on the error value).

Slow steering compensation methods

A tractor usually has hydraulic steering which is slow compared to the reaction a light servo motor has.
The speed of steering can be approximated to a first order system (Vougioukas, 2012) as:

$$\dot{\varphi} = -\frac{1}{\tau_\varphi}\varphi + \frac{1}{\tau_\varphi}u_\varphi$$
(5)

Where φ and $\dot{\varphi}$ are the steering angle and steering motion speeds, τ_{φ} the lag of the steering system, u_{φ} is the desired steering angle.

A slow steering reaction significantly alters the control paradigm and can result in large overshoot. The servo steering system is approximated on the robot with a ‘steerslow’ command. Students are tested with tuning the navigation control to work with a slow steer command compensating for this and giving them the experience of the problems of implementing onto a hydraulic system.

Maintain velocity

Velocity can be read through the hall sensors on the wheel which delivers pulses as it rotates. For a relatively slow wheel with few pulses per revolution, the velocity is calculated from the time between pulses. The velocity is:

$$V_{m/s} = \frac{circ}{Np \times t} \qquad (6)$$

Where Np is the number of pulses per revolution, t is the time measured between pulses (sec), and circ is the wheel circumference.

For a fast moving wheel with many pulses per revolution, it is more efficient to count the number of pulses for a fixed time, in which case, the velocity is:

$$V_{m/s} = \frac{pulses \times circ}{time \times Np}$$
(7)

Where $pulses$ is the number of pulses counted in a specific timed interrupt, $time$.

A clever algorithm could even switch between the two methods, continuously comparing the answers as, due to time and pulse quantisation, there will always be slight errors.

The task, to maintain velocity using the control loop in Figure 5 necessitates a PI controller. Students can explore the importance of integration with velocity control as proportional only control will result in a steady-state velocity error.

GPS and coordinate transformation

The GPS receiver reads the latitude and longitude to 6 decimal places. With the robot having a clear view of the sky, students can see the position values on the LCD settle towards a more accurate value, the properties of fluctuation, shadow from buildings, and the time it takes to get a fix are observed. Moving outside and viewing the coordinates gives a natural feel for the directions and the resolution of these values and how they are affected by position. Typing these values into Google Earth always gets an interest when it shows their location. The inaccuracy of the GPS can be seen and students think about how this affects navigation.

Universal Transverse Mercator

UTM is a system which converts the coordinates into metres. Whereas latitude and longitude are measured in degrees, the conversion to metres of Northing and Easting allows trigonometry and the Pythagoras rule to be used to calculate bearing and distance. Students can walk North and East and see the points changing as well as calculate the bearing and distance to certain pre-defined points. There is usually some landmark or item at these points-giving the students a sense of achievement when getting there. The calculated bearing can be compared with the compass bearing to see the requirement for calibration and taking the effect of the deviation of magnetic bearing.

Navigation

Waypoint navigation relies on knowing the waypoint position, the current robot position, and the current robot heading. Remembering that GPS coordinates are simply points without heading, the direction of the waypoint from the position of the robot is calculated (figure 5). This uses trigonometry by taking the UTM values of Northing and Easting points of the robot and the waypoint and using the tan-1 function to calculate the angle between them. This relies on the computer function ‘ATAN2’ which computes the full 0-359º heading rather than the limited 0-180º in standard use. Referring to Figure. 5, this angle is the desired heading (Input), which the robot should head towards. The compass, (Sensor), is used to determine the robot’s current heading (Output), and the control algorithm developed earlier takes the difference between the desired and actual heading (error), to calculate a steering angle.

The distance to the waypoint uses Pythagoras’s theorem and controls the drive speed of the robot (up to its limits, or the desired speed of the farming operation).

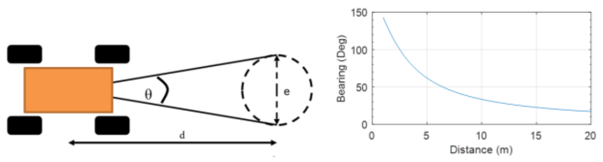

Figure 5. Bearing error values approaching a waypoint

The robot should stop when it is within a minimum distance to the waypoint due to the inaccuracy of the GPS interfering with the bearing calculation at a close distance. This can be explained that as the robot approaches closer to a waypoint, the calculated bearing becomes less accurate and bearing flutter increases as the proximity decreases with the relationship:

$$\theta = 2\tan^{-1}\left(\frac{e/2}{d}\right) \quad (8)$$

where θ is the maximum bearing error, e-the maximum GPS error, and d-the distance to the waypoint. A graph shown in Figure. 7 plotting this relationship allows the student to think about how inaccuracies affect navigation. If the algorithm of the robot was to try to only stop when it was exactly at the waypoint, this could result in a circling of the waypoint with the robot on full steering lock.

State programming and behaviours

Students are encouraged to write their own program to navigate to a waypoint, or multitude of waypoints (defining a path) and to avoid obstacles. As programs increases in complexity, a collection these higher level tasks can be encapsulated in a 'behaviour', the concept of which is introduced a few days into the course.

Behaviours are a collection of higher level actions the robot performs, such as following a wall a certain distance, or navigating to a waypoint. This usually involve multiple inputs and outputs and a control loop in-between. Whilst it can be advantageous for students to develop their own behaviour programming, they soon start to see the need to encapsulate these to ensure the program remains concise and understandable. For example, 'Drive target' controls the steering only of the vehicle to face it towards a target position. The function returns the distance to the target and this can be used for speed control.

More complex behaviours can be built from simple behaviours, for example, path following, this could be used for example to get the robot from a garage to the start of a field along a path. A path can be defined as a collection of waypoints, and a program allows for a stack of waypoints to be defined with a bounding radius. This radius has to be greater than the accuracy of the GPS receiver otherwise the robot could rotate continuously about a waypoint without reaching it as mentioned in the previous section.

Although behaviours can be written to achieve a specific task, e.g. drive and turn, drive to a location, etc., students often struggle with putting it all together, e.g. if the distance to an object is small, turn and follow the side of it, these behaviours can be selected by the use of 'States' (figure 6).

States

The 'state' of a robot is the reason it is performing a particular behaviour, and the reasons are usually driven by an external sensor or internal timer. For example, the state of a robot may be causing its behaviour to drive towards a waypoint, however, the ultrasonic sensor detects an obstacle in its path and so it switches its behaviour to avoid the object by looping around it.

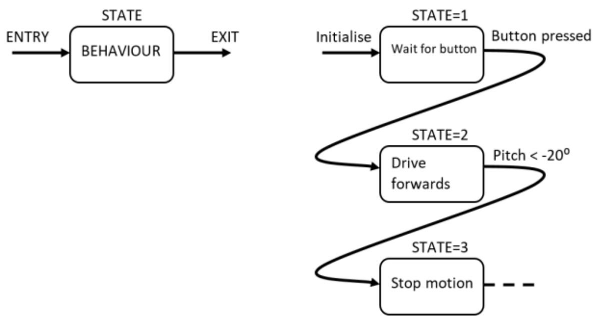

Figure 6. State diagrams

Each state has an Entry Point, Behaviour, and Exit point, the active state is represented by a number, which is associated with a particular behaviour. An example is shown in Figure. 6 where the robot initially is in State=1 and the behaviour is to wait for a button press. Once pressed, the state value increases and its behaviour is to drive forwards. Once the pitch sensor detects it is on a downward slope of less than 20º, the state increases and the robot stops.

Navigation

Two types of navigation are considered: Straight to Waypoint (STWP) means the robot navigates to a waypoint directly, and Minimise Cross-Track Error (MXTE) where the robot attempts to follow a line to a goal point (defined by two points A-B). Several studies have attempted to calculate the gains necessary to compensate for the goal bearing along with the line offset, a 'follow the carrot' algorithm, where the 'Aim point' is a constantly moving point ahead of the vehicle along a 'slot' of the navigation line (A-B line).

This point is constantly moving in-front of the robot by an 'aim distance'. This 'slot-car' approach simplifies the navigation to follow a straight line. The Aim point position (D) is calculated as follows:

Calculate deviation from the desired trajectory line (FP):

$$FP = \sqrt{AP^2 - AF^2} \quad (9)$$

where:$AP^2 = (xg - xa)^2 + (yg - ya)^2$ and (xa, ya) is the line start point (A) and (xg, yg) is the final goal point (B). Find AF:

$$AF = AP \cdot \frac{AB}{|AB|}$$

$$AF = AP \cdot \frac{(xa - xb), (ya - yb)}{\sqrt{(xa - xb)^2 + (ya - yb)^2}}$$

$$AF = \frac{(xa-xg)*(xa-xb)+(ya-yg)*(ya-yb)}{\sqrt{(xa-xb)^2+(ya-yb)^2}} \quad (10)$$

To find the coordinates of the perpendicular point F (fx, fy), we know the equation of the line A-B, and the distance along it, AF, so:

$$B = \sqrt{(xa - xb)^2 - (ya - yb)^2}$$

$$F_x = xa + (xb - xa) + \frac{AF}{AB} \tag{11}$$

$$F_y = ya + (yb - ya) + \frac{AF}{AB} \tag{12}$$

Then, the displacement from the path (FP) as previously calculated along with knowing the perpendicular point F on the path (A-B) are use to find an 'aim point'. This is a point, some way ahead of the ideal point on the path where the robot should be D(x,y) and is given as a reasonable point on the ideal path for the robot to aim for. From this, δ_{aim} is calculated and used to control the rotation of the robot.

$$D_x = F_x + (xb - xa) + \frac{Aim\ dist}{AB}$$

$$D_y = F_y + (yb - ya) + \frac{Aim\ dist}{AB}$$

$$\delta_{aim} = atan2\left(\frac{D_y - y}{D_x - x}\right) \tag{13}$$

Where Aim dist is a value fixed at a reasonable distance ahead of the vehicle, e.g. 5 metres. All coordinates are in UTM metres (figure 7).

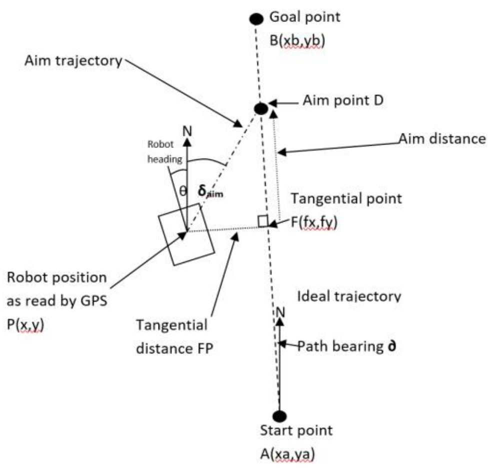

Figure 7. Minimising cross-track error

4. RESULTS

Levels of learning

The levels of engagement and thinking can be paralleled with Bloom's Taxonomy, the basic levels of Knowledge and Comprehension are learned through the use of keywords in the program, quickly moving to Application once they are applied in a useful program, see Table 3. Students choosing which sensors to use for a particular task allows Analysis and then leading to Evaluation of the task, and Synthesis to understand the actions and redesign the solution.

While moving up through the Taxonomy levels demonstrates an increase in Complexity, Difficulty is the amount of effort at each level, e.g. increasing the Difficulty in Comprehension of digital I/O would be to ask a student to create several LEDs to blink in time rather than one LED, the task is not more complicated but is more time consuming. The goal is to have students working at the top levels of the taxonomy as quickly as possible. However, as a program becomes more difficult (more keywords and sensors working), it can swamp the higher levels of learning (students become confused with the basics). Hence, some keywords are used to encapsulate higher levels of robot behaviour and present them as simple knowledge keywords, e.g. 'Steer_to_point' is a keyword which takes in a goal location, reads the on-board GPS and the orientation, calculates the desired bearing to the goal, and controls the steering to point in the correct direction. These behaviours are only introduced later on (so the student isn't tempted to short-circuit learning the basics-but to understand the concepts), and these can free the mind to develop higher order control based on existing behaviours.

The workshops are very popular and many institutions want to send their students to participate (figure 8).

A Smallpeice Trust (https://smallpeicetrust.org.uk/) residential has been running every year since 2016 over three days focussing on 16-17 year olds and caters for 20 students, this has resulted in recruiting on average an extra two students annually into our University Engineering courses (we recruit in total approx. 60 each year). A three-week summer school attracting students from three universities in China has been running since 2013 where 40 undergraduates study robotics coupled with English classes and educational trips. This has resulted in a collaboration agreement where we run a Master's degree in Applied Mechatronic Engineering at our University for their students. The same robot hardware has been used to teach programming basics to home educated children with age ranges from 8-18 in a single afternoon.

The robot system is used in Canada, the Philippines, and India to deliver courses in robotics both face to face and remotely.

Work is progressing to embed the components into a single PCB, include a simple vision system, improve communications using LoRaWAN, and integration with a mobile phone as a terminal device using Bluetooth (this could lead to further programming of the mobile device for a user-friendly GUI).

Algorithm developments are continuing in the area of navigation and path following and this system has been particularly useful for use in the development and testing of such software idea and approaches.

Students like to demonstrate what they've done, and an obstacle course was created to form a competition. This consists of a variety of obstacles which required the use of various sensors to detect. Each obstacle is progressively difficult to navigate and so this is judged on how far the robot can progress through the course and how well it avoids obstacles.

Table 3. Bloom's Taxonomy applied to learning using a robot

Complexity→more complicated tasks	**Taxonomy**	**Example of student work**	**Example keyword or application**
	Evaluation	Redesigning the program parameters for the task	Behaviours
	Synthesis	Creating behaviours for the robot	drive_to_point();
	Analysis	Choosing the sensors and actuators for the task	read_ultrasonic read_lat
	Application	Using the keywords in a program to achieve a task	if (read_button(1)) led_out(1,HIGH);
	Comprehension	Adapting the inputs to the keywords for a task	led_out(1,HIGH);
	Knowledge		led_out
	Difficulty→increasing the volume at this level		

Figure 8. Typical workshop at University of Santo Tomas, Philippines

5. CONCLUSIONS

Learning is encouraged by application, the hunger for extra knowledge is generated by the student when they have taken their current steps of knowledge and see positive results. Small steps of learning through the testing of the program code with the robot are amalgamated into larger applications where greater steps of achievement can be taken.

The fundamental ideas of control theory are strengthened with experiments demonstrating the power of feedback control, this is a good test-bed for students to test and apply their knowledge, particularly in learning how to apply feedback in software and the effects of tuning.

Complex concepts are broken down into small, achievable tasks, covering the basics and encouraging self-checking and peer group work. The practical exercises are defined to limit the time spent at each level of learning, with each subsequent level being progressed once the student has grasped the concepts by experiencing their algorithms come to life. The library repository, schematic diagrams and presentations are available here https://github.com/swane/atlas.

References:

Barker, B. S. (2012). *Robots in K-12 education: A new technology for learning*. Information Science Reference.

Bautista, A. J., & Wane, S. O. (2018). ATLAS Robot: A teaching tool for autonomous agricultural mobile robotics. *Proceedings of the 7th International Conference on Control, Automation, and Information Sciences (ICCAIS 2018)*, 264-269. https://doi.org/10.1109/ICCAIS.2018.8570494

Blackmore, S., Apostolidi, K., & Fountas, S. (2010). FutureFarm: Addressing the needs of the European farm of the future: Findings of the first two years. *IFAC Proceedings Volumes*, *43*(26), 1-17. https://doi.org/10.3182/20101206-3-JP-3009.00003

Build the Bridge - Play it now at Coolmath-Games.com. (n.d.). Retrieved April 5, 2018, from http://www.coolmath-games.com/0-build-the-bridge

Coordinate conversions made easy. (n.d.). Retrieved April 11, 2018, from https://www.ibm.com/developerworks/library/j-coordconvert/

Global Citizen - Join the movement changing the world. (n.d.). Retrieved September 10, 2023, from https://www.globalcitizen.org/en/

Hands Free Farm - Home. (n.d.). Retrieved September 10, 2023, from https://www.handsfree.farm/

Hands free hectare - Home. (n.d.). Retrieved September 10, 2023, from https://www.handsfreehectare.co.uk/

Harper Adams University - Home. (n.d.). Retrieved September 17, 2023, from https://www.harper-adams.ac.uk/

John Deere is buying an AI startup to help teach its tractors how to farm. (2017). *The Verge*. Retrieved March 7, 2018, from https://www.theverge.com/2017/9/7/16267962/automated-farming-john-deere-buys-blue-river-technology

Kulich, M., Chudoba, J., Kosnar, K., Krajnik, T., Faigl, J., & Preucil, L. (2013). SyRoTek-Distance teaching of mobile robotics. *IEEE Transactions on Education*, *56*(1), 18-23. https://doi.org/10.1109/TE.2012.2224867

Pittsburgh (2016). Helping students build conceptual models - The lost manual. Carnegie Mellon Robotics Academy. Retrieved March 7, 2018, from https://s3.amazonaws.com/cs2n/research/The%20Lost%20Manual%20MEA.pdf

Shoop, R., & Flot, J. (2016). Can computational thinking practices be taught in robotics classrooms? Why teach computer science and computational thinking in robotics class? *Proceedings of the 2016 Robotics Education Conference*.

Vougioukas, S. G. (2012). A distributed control framework for motion coordination of teams of autonomous agricultural vehicles. *Biosystems Engineering, 113*(3), 284-297. https://doi.org/10.1016/j.biosystemseng.2012.08.013

We Robotics. (n.d.). Retrieved March 7, 2018, from https://werobotics.org/

World population projected to reach 9.7 billion by 2050. (2015). United Nations Department of Economic and Social Affairs. Retrieved March 7, 2018, from http://www.un.org/en/development/desa/news/population/2015-report.html

Samuel Oliver Wane
Harper Adams University
United Kingdom
swane@harper-adams.ac.uk
ORCID 0000-0003-3810-2902

Matthew Butler
Harper Adams University
United Kingdom
mbutler@harper-adams.ac.uk
ORCID 0000-0003-3026-8465

Anthony James Bautista
University of Santo Tomas
Philippines
acbautista@ust.edu.ph
ORCID 0009-0000-8686-4712