PID Based Design and Development of a Mobile Robot Using Microcontroller

Chapter · July 2018

DOI: 10.1007/978-3-319-95165-2_49

7 authors, including:

Mukhtar Fatihu Hamza
Prince Sattam bin Abdulaziz University
85 PUBLICATIONS **764** CITATIONS
SEE PROFILE

Aminu Yahaya Zimit
Universiti Teknologi Malaysia
12 PUBLICATIONS **94** CITATIONS
SEE PROFILE

Sani Danjuma
Northwest University, Kano
17 PUBLICATIONS **148** CITATIONS
SEE PROFILE

ROHADI Erfan
Politeknik Negeri Malang
118 PUBLICATIONS **446** CITATIONS
SEE PROFILE

PID Based Design and Development of a Mobile Robot Using Microcontroller

Mukhtar Fatihu Hamza[1,2](✉), Joshua Lee Zhiyung[2], Aminu Yahaya Zimit[1,2], Sani Danjuma[3], Erfan Rohadi[4], Silfia Andini[5], and Tutut Herawan[4,6,7]

[1] Department of Mechatronics Engineering, Bayero University, Kano 3011, Nigeria
emukhtarfah@gmail.com
[2] Department of Mechanical Engineering, University of Malaya, Kuala Lumpur, Malaysia
[3] Department of Computer Science, Northwest University, Kano, Nigeria
[4] State Polytechnic of Malang, Malang, Indonesia
[5] Universitas Putra Indonesia YPTK, Padang, Sumatera Barat, Indonesia
[6] Universitas Negeri Yogyakarta, Yogyakarta, Indonesia
[7] Universitas Teknologi Yogyakarta, Yogyakarta, Indonesia

Abstract. Human labor work has been increasingly being replaced by robots to do works for decades, especially those that are tedious and risky. Moreover, works done by robotics have less error prone and higher efficiency. This paper presents a design and develops microcontroller based mobile robot (MR). The robot is based on a microcontroller, acting as the brain, which contain a series of programs that interpret its surrounding through input data from the sensors and maneuver through the limited data obtained while avoiding obstacle. The prototype was able to differentiate surrounding and maneuver to desired location smoothly according to predefined path while simultaneously sensing for obstacle to avoid. The Proportional–Integral–Derivative (PID) based program on the microcontroller was critical because it needs to handle the sensitive sensors feedback and sent correct command to respond to the surrounding while at the same time having the correct arrangement and format. The developed MR can be controlled autonomously, following the path by varying the current fed to the motor through error correction, towards desired location while simultaneously sensing for obstacle as far as 400 cm ahead. Through decision making the speed of the MR will adjust itself and will put stop moving when the obstacle was 5 cm ahead. Based on the experiment the advantage and disadvantage of the current development were realized for further development.

Keywords: Mobile robot · Microcontroller · PID controller · Sensor

1 Introduction

Over the years, development of the human society has been greatly improved with robotics as a dominant contributor. It is a field that requires combination of effort of a variety of scientific areas such as mechanical, electrical, control and software engineers [1, 2]. The mobile robot (MR) was firstly brought to market in 1950's by Barrett Electronics Corp (Northbrook, IL) and at the time it was simply a tow truck that follows a wire in the floor instead of a rail [3]. The earliest guided MR is line following MR. This is a type of MR that can follow a specific predetermined path by the user which acting as a guidance device where it can be simple physical line on the floor or complex invisible lines like a magnetic field embedded on the travelling horizon. This means the robot has the ability to follow path by sensing it and manoeuvre itself to stay on predefined path, correcting wrong moves constantly through embedded feedback mechanism to travel to its destination [4]. This type of MR may vary from simple low cost line sensing circuit to expansive vision systems. Although they may not be glamorous of robots, but their works are often essential to smooth running of factories, offices, hospitals and even houses. The usage of line following robot (LFR) is increasing day by day. From industrial point of view LFR has been implemented in semi and fully autonomous plants. The application in these environments usually functions as materials carrier to deliver products from one point to another where rail or conveyor solutions are not very suitable. The automation is very much desired due to the capability to always run non-stop while not getting tired in all sorts of environment as well as accelerate automatic transportation without any complaint [5, 6].

Punetha *et al*. developed MR in health care management system [7]. They described the techniques for analysing, designing, controlling and improving the health care management system using a LFR. In another side of health care services, MR has been proposed to be elderly-care. This research is done by Takuma *et al*. [8]. They proposed a MR that can detect human fall and respond by reporting to observers. Singh *et al*. [9] introduces a cell-phone detection based LFR. They presented a real time detection of mobile phone in restricted area using MR. The LFR uses mobile transmission detector to sense the presence of activated mobile phone from distance of one and a half metre and the robot will travel and stop at the location of the activated mobile phone. There are many reasons which yield to the creation of MR around the world. Most of them are to overcome the logistic problems that often occur in workplaces and to make improvement to the facilities provided in the workplaces [10]. In industries or factories, they can ease the physical strain on human workers by performing tiring tasks, such as lifting heavy materials, more efficiently with no signs of fatigue creeping in or dealing with continuous repetition of works without getting tired [4]. They can endure greater work load than human workers and their movements can be tracked and timed at all times. Moreover, they are being developed to be autonomous and being applied to be fit and forget as little supervision needed on them [10]. As a result, MRs have been increasingly replacing human labour to do their works. However, these MRs are not universal and bounded with limitations. Different types of MRs have been developed according to the working environment and their required work done. Meanwhile MRs are very well developed in industrial field and the like with similar environment, they are not very well developed in agricultural field. This is due to agriculture field having a completely different environment than factories and job scope. Thus, a study is needed to apply MRs to agricultural environment to perform works. This states the motivation of this work. Furthermore, the MR should always be in stable and functional condition.

This study develops a stable and useful guided MR, with proper study and accurate model regarding electronics and steering mechanism. After model design and improving performance, proper stability analysis was presented in this work. The line sensing process of the developed MR has a high resolution and high robustness. Despite the complexity of the LFR, the developed MR has the capability to sense tags to perform works accordingly, navigate junctions and decide or which junction to turn/ignore or having requirement to perform a 90° turn and also junction counting capabilities.

The rest of this paper is organized as follow: Sect. 2 presents proposed method. Section 3 presents obtained results and following by discussion. Finally, Sect. 4 concludes this work.

2 Proposed Method

Any embedded application generally involves a number of functions. The proposed smart and intelligent LFR consists of few basic parts which are sensors, comparator and actuators. Common LFRs use reflective optical infrared (IR) sensors to sense line which consist of an array of diodes in pair which one of them, light emitting diode (LED), sends ray while the other, light dependent resistor (LDR), receive the reflection ray. The output of the sensors is an analogue signal which depends on the amount of light reflected back to LDR due to the varying of the LDR resistance. The signal is given to the comparator to translate into digital binary system language which then fed to the logic circuit whose output is then passed through a diode matrix and then given to the driver circuits and finally generating instructions to driving motors. In this project, similar system is used, with the modification of orientation of the sensors, using raw data to differentiate colour and wireless output module for data collection. In addition,

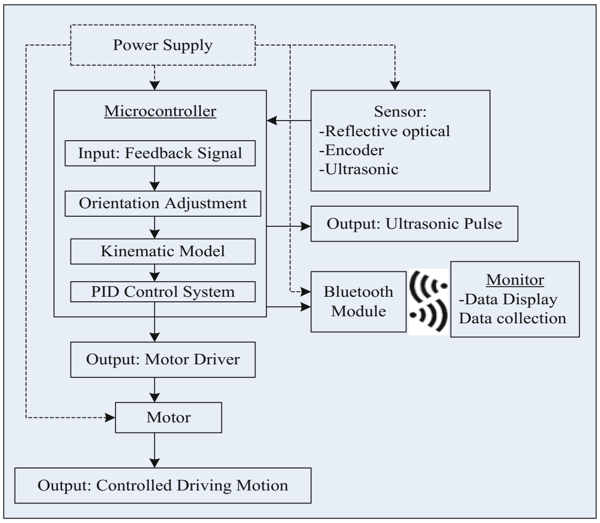


Fig. 1. Proposed system flow

ultrasonic sensor is added as added feature for obstacle detection and executes collision avoidance and rerouting the MR. The proposed system flow is shown in Fig. 1.

2.1 PID Controller

The main controlling system of the propose MR adopt PID control. The MR uses sensor feedback data as PID control variable to calculate an output response to do correction and follow the predefined path. The equation is given as follow [11]:

$$Output = P + I + D = K_p e(t) + K_i \int e(t)dt + K_d \frac{d}{dt} e(t) \tag{1}$$

where P is proportional term which accounts for present error.

$$K_p e(t) = K_p * (\text{setpoint} - \text{input data}) \tag{2}$$

The I is integrate term that accounts the total error history.

$$K_i \int e(t)dt = K_i[(\text{error})_1 + (\text{error})_2 + (\text{error})_3 + \ldots \tag{3}$$

The D stands for derivative which accounts for future error through differential/rate changes.

$$\text{differential error} = K_d * \left[\frac{\partial}{\partial t}\text{error}\right] \tag{4}$$

The K_p, K_i and K_d denotes the coefficients of the proportional, integral and derivative terms respectively. This equation is computed frequently through micro-controller at a very high frequency. Thus several addresses need to be made such as sampling time, derivative error, tuning, reset windup and on/off.

2.2 Sampling Time

The problem with running a PID through microcontroller is that the function will be called irregularly. This usually causes inconsistence behaviour and the MR will not be stable. Thus extra math needed in computing the time dependent values, derivative and integral. As such, following calculation added into the PID calculation [12].

$$K_i = K_i(\partial \text{time}) \tag{5}$$

$$K_d = \frac{K_d}{\partial \text{time}} \tag{6}$$

2.3 Debugging Derivative Error

Whenever there is a change in set point, especially during starting point from zero to a set point value, there will be a disturbance in the derivative, which causes instantaneous change in error, end up a very big undesired number. This can be debugged by removing the set point variable from the derivative equation [13].

$$\frac{\partial}{\partial t} error = \frac{\partial setpoint}{\partial t} - \frac{\partial input}{\partial t} = -\frac{\partial input}{\partial t} \tag{7}$$

2.4 Tuning

It's crucial to be able to tune the PID coefficient to suits different situations. The PID will go wild when there are changes in parameter. The integral coefficient involves the summation of error history, a tune in K_i results in changes in the error history. Thus, modification of the calculation done by including each K_i into the integral instead of outside the integral [12].

$$K_i \int e(t)dt = (K_i * \text{error})_1 + (K_i * \text{error})_2 + (K_i * \text{error})_3 + \ldots \tag{8}$$

where the desired yaw rate,

$$\gamma_d = \omega = \frac{V_o(1+S_x) - V_i(1+S_x)}{d} \tag{9}$$

$$R_{\text{actual}} = \frac{V_o(1+S_x) + V_i(1+S_x)}{V_o(1+S_x) - V_i(1+S_x)} \cdot \frac{d}{2} \tag{10}$$

while tuning the sampling time, the integral and differential coefficient scaled. Thus the coefficient needs to be resized by either multiplying or dividing according to the ratio of sampling time [13].

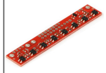
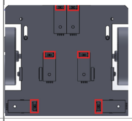
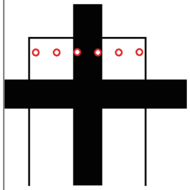
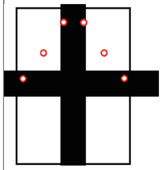
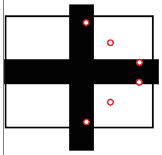
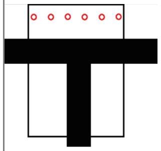
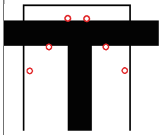
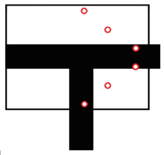
$$K_i = K_i(\partial \text{sample time}) \tag{11}$$

$$K_d = \frac{K_d}{\partial \text{sample time}} \tag{12}$$

2.5 Reset Windup

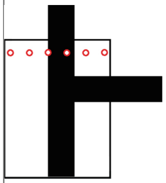
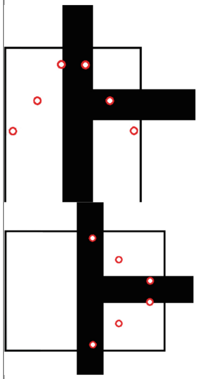
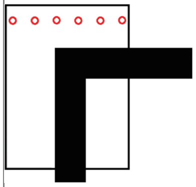
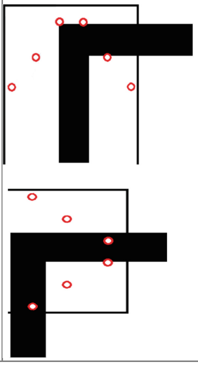
The microcontroller has no boundary, thus if the PID function runs incrementally, then at some point it will reach point which doesn't compatible with the hardware. Arduino PWM output ranged from 0–255 whereas the microcontroller has no boundary thus it will consider value higher than 255. This will results an unexpected lags in program and damage the microcontroller ports. The solution is to clamp the final output value to be at its max when it's over or min when it's lower by using microcontroller control structure.

Table 1. Comparison between conventional configuration and propose designed configuration

Situation	Conventional	Prototype	Explanation
			Sample design configuration
Cross junction		 	Sensor 1, 3/4, 6 detects paths, indicating it reached the centre of the cross junction and the control shifted towards decision making in either resuming straight path or self-rotate towards either left or right before resuming PID control. If the MR decision is to turn at the junction, the self-rotation will function until the sensor 1 and 6 detects the path again, indicating that the MR manoeuvred to the left or right path direction. Thus, PID can take over and resume its control over the MR. The conventional method however rely on either time delay which accuracy reduces with drop of battery charges or additional hardware to ratify it reaches the centre of the junction
T-junction (Minor road)		 	Sensor 2, 3, 4 and 5 detects path, indicating it reaches a split junction, while sensor 2 and 5 still detects path, eventually sensor 3 and 4 will not detect any path as the MR moving forward. This indicate that the MR approaching a T-junction, at the side of minor road, instead of a cross junction. The PID control edge sensing still functions with edge now shifted towards sensor 2 and 5, and later on 1 and 6, until the centre of the MR reaches the split point. Thus the control can be shifted to decision making on path selection and initiate self-rotation until the sensor 1 or 6 detects path, indicating the MR has reached the direction of T-junction main road. At initiation of PID, sensor 1 or 6 is set to temporary removed from the equation for PID to function normally. In conventional design, the sensors face the similar problem with cross junction. In addition, the sensors totally unable to rely on PID control as it approaching the

(*continued*)

Table 1. (*continued*)

Situation	Conventional	Prototype	Explanation
			junction as all the sensors unable to detect any path. This made the PID function unable to control the MR to approach the turning point. Alternative method needed to overcome this issue, with additional parts as a prerequisite
T-junction (main road)			The concept is similar to cross junction. Instead of detecting both side, it only detect one side, indicating the MR approaching a T-junction with the MR at the main road of the T-junction. In this condition, the PID function normally while the sensor 5 is to be temporary subtracted out of equation. When either sensor 1 or 6 detects, PID control is swapped with to decision making of either turning or continue following the main road. When the decision is to manoeuvre towards minor road, the self-rotate initiate and ends in the exact same method as cross junction. Therefore, simplifying the decision making program
Corner			In this situation, the concept adopts the combination of T-junction main and side road concept in detecting the corner path. In this condition, the program straight away jumps to self-rotating function and resumes PID control once sensor 3, 4 and 1 or 6 sensed the predefined path. While initiation of the PID, sensor 1 or 6 temporary removed from the equation

2.6 PID On/Off

At certain point the condition requires closing the PID control, such as the occurrence of multiple lines and junctions. When it resumes, the PID will have an unexpected bounce due to the integral and derivative equation. Thus when the PID equation needs to reset the integral and the derivative resets.

$$\partial \text{Input} = 0 \tag{13}$$

$$K_i \int e(t)dt = 0 \tag{14}$$

The PID equations are converted into Arduino code as a type function to be called, as shown in Appendix 1. The configurations of the sensors are carefully designed to enable detection junctions in addition of having conventional line sensing. Table 1 shows the comparison between conventional configuration and currently designed configuration at different situations, where the red dots represent single unit optical reflective IR sensors, labelled as sensor 1 to 6 from left to right, and the rectangle represent the MR body.

2.7 MR Design

The design focus more on the intelligence of the MR, mostly based on feedback system, embedded into a microcontroller [14, 15]. The design starts with the architect of the system to decide the parts and material needed. When the parts finalized, CAD software SolidWorks, as shown in Fig. 2, without wiring, used to model the MR for virtual inspection of the orientation and assembly. The MR modified from time to time until the final design is confirm.

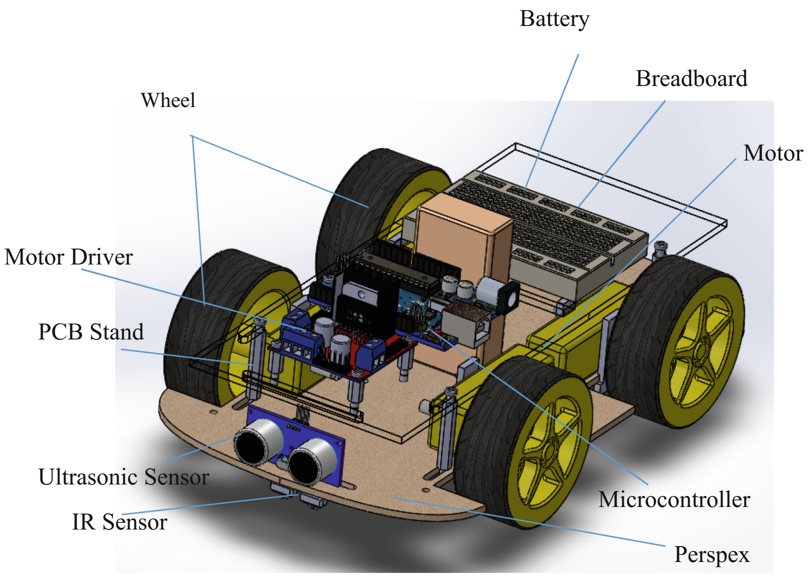


Fig. 2. Mobile robot CAD modelling

2.8 Communication Method

Stable and correct communication is needed especially when dealing with an operating MR [15]. Improper communication method would lead to data missing or programming standstill. The communication protocol involves both sender and receiver. The initial command starts from keyboard sending data to Arduino UNO which runs the function once using serial communication, as shown in Appendix 2, through Bluetooth module. The function is then while looped until reached destination or interrupted by central command, as shown in Appendix 3, when the serial communication is available.

Tuning parameter while operating is essential, to instantly fixing any unexpected change in environment. Thus, when entered the tuning function, the microcontroller will ask for new setting and stall until the there is an input of new parameter by an empty conditional loop as shown in Appendix 4.

Once the model is finalized, the process proceeds to fabrication. A prototype model fabricated shown in Fig. 3.

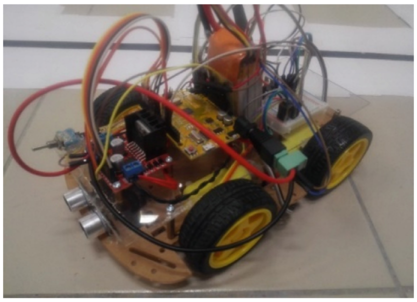


Fig. 3. Prototype model

3 Results and Discussion

After the completion of prototype fabrication, further development will involve the data extraction and computing reaction based on extracted data. The testing conducted breaks into several response starting with PID control, obstacle avoidance function and path selection. Since the MR, programming and setting made by hand on top of the limiting budget, the movement of the MR might have functional error and stability issues. However, the errors are covered by utilizing the microcontroller available software capacity. The whole program sketch thus far used 11,302 bytes (35%) of program storage space, out of maximum of 32,256 bytes and 547 bytes (26%) of dynamic memory for global variables, leaving 1,501 bytes for local variables. In order to design an autonomous MR, the environment must first be predefined. The testing field was designed according to a standard grid as shown in Fig. 4.

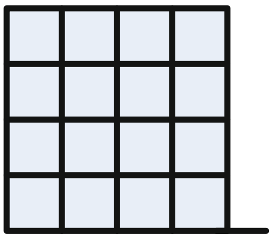


Fig. 4. Testing field

This test involves testing the MR PID program functionality in maneuvering according to predefined line. The test starts with the reaction test of the reflective optical IR sensors, placing the path under third and fourth sensor, and extraction of sensors feedback data which displayed in serial monitor to set the comparator program

Table 2. PID control result

Set point	Input	PID output	Error	Total error	Differential error	Left PWM	Right PWM
3.5	6	−217.49	−2.5	0.01	0.5	220	3
3.5	6	−219.99	−2.5	0.01	0	220	1
3.5	5.5	−178.5	−2	0	−0.5	220	42
3.5	5.5	−176	−2	0	0	220	44
3.5	5	−134.5	−1.5	0	−0.5	220	86
3.5	5	−132	−1.5	0	0	220	88
3.5	4.5	−90.5	−1	0	−0.5	220	130
3.5	4.5	−88	−1	0	0	220	132
3.5	4	−46.5	−0.5	0	−0.5	220	174
3.5	3.5	−2.5	0	0	−0.5	220	218
3.5	3.5	0	0	0	0	220	220
3.5	3	41.5	0.5	0	−0.5	179	220
3.5	3	44	0.5	0	0	177	220
3.5	2	127	1.5	0	−1	94	220
3.5	2	132	1.5	0	0	89	220
3.5	1.5	173.5	2	0	−0.5	47	220
3.5	1.5	176	2	0	0	45	220
3.5	1	217.5	2.5	0	−0.5	3	220
3.5	3	54	0.5	0	2	166	220
3.5	3	44	0.5	0	0	176	220
3.5	1	210	2.5	0	−2	10	220
3.5	1	220	2.5	0	0	0	220
3.5	1.5	178.5	2	0	0.5	42	220
3.5	1.5	176	2	0	0	44	220
3.5	1	217.5	2.5	0	−0.5	3	220
3.5	1.5	178.5	2	0	0.5	42	220
3.5	1	217.5	2.5	0	−0.5	3	220
3.5	2.33	109.34	1.17	0	1.33	111	220
3.5	2.5	88.84	1	0.01	0.17	132	220
3.5	1	212.51	2.5	0.01	−1.5	8	220
3.5	1	220	2.5	0.01	0	0	220

parameter, as Appendix 5. The data shows that the data fluctuate below 220 when sensing white region and above 800 when sensing the black path. Thus the comparator program can be defined as shown in Appendix 6. Afterwards, all the variable of the PID equation is verified and compared, as shown in Appendix 7, through serial monitor display, with the mathematical calculation. Table 2 which is extracted through processing program as shown in Appendix 8-show the results of deviation and PID response controlling the left and right motors of the MR.

In physical test, initially the MR placed in an orientation deviate from the center line before initiation, as shown in Fig. 5(a). Through PID calculation the MR detects the deviation and steer itself towards the center line, as shown in Fig. 5. The test was repeated with several different deviation orientations. The program is review and updated to debug unusual behavior of the MR, mainly due to non-precise handmade prototype and cheap standard part physical defect.

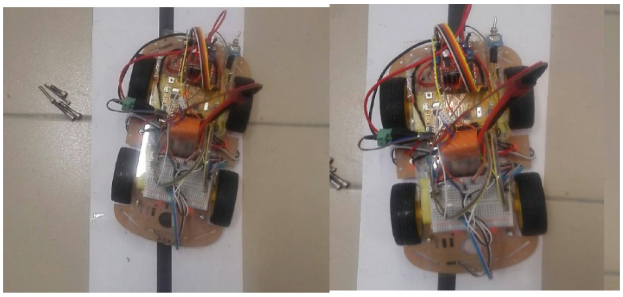


Fig. 5. MR (a) Before and (b) After initiation

3.1 Obstacle Avoidance

This test involves testing the MR proximity reaction. The test starts with the reaction test of the ultrasonic sensors. Obstacle placed twenty centimeters in front of the MR and moving towards it. The sensor feedback data extracted and displayed in the serial monitor, as shown in Fig. 6, together with post processed data in unit centimeter. A graph of actual obstacle distance against sensor input distance plotted as shown in Appendix 9. The graph compares the actual distance and distance computed by microcontroller based on ultrasonic sensor feedback data. It is shown that the computed data is quite stable with a slight ±1 cm fluctuation. At the end it is shown that there is a limit towards accuracy as starts from 2 cm and nearer the computation went wild. After that, test run is conducted to observe the relationship between proximity function and the PID response. Figure 7(a) show the MR still collide even though it sensed obstacle ahead before stopping. This is due to the delay in reaction time from looping a long list of program. Thus timer interrupt function, which interrupt the loop in a constant 50 μs, to check for obstacle is added for improvement. A success in avoiding collision is shown in Fig. 7(b).

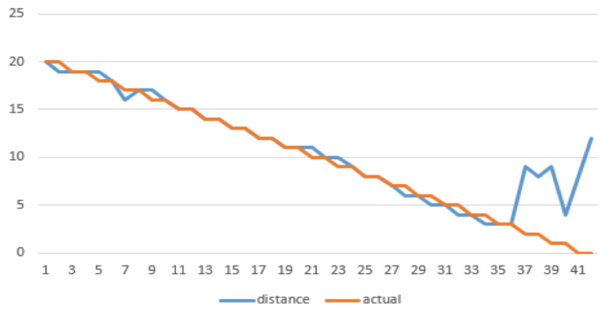
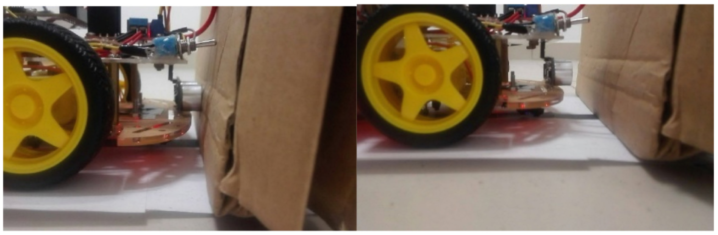


Fig. 6. Comparison between actual distance and sensor computed distance



(a) (b)

Fig. 7. MR halt position (a) without and (b) with timer interrupt

3.2 Path Selection

The final destination is predefined in central command and tested in testing field. The MR is set to turn left on first junction, then turn right on the third junction before reaching final junction as destination using PID control on straight path and junction selection at junctions. The result shows that the MR able to reach its destination correctly, as shown in Fig. 8.

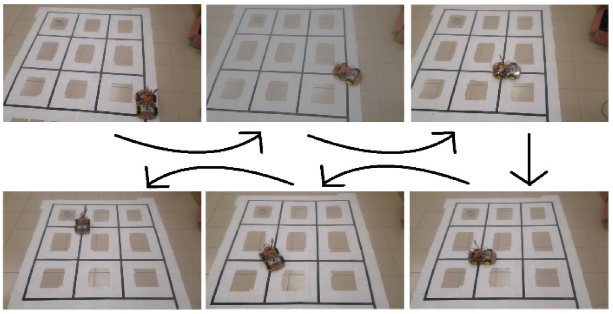


Fig. 8. MR field test

3.3 Colour Sensing

A test is conducted for the reflective optical IR sensor to sense red, green, blue (RGB), Cyan, Yellow and Magenta (CYM) colour. The range for each colour is determined and a graph, as shown in Fig. 9, of the feedback data against time is plotted.

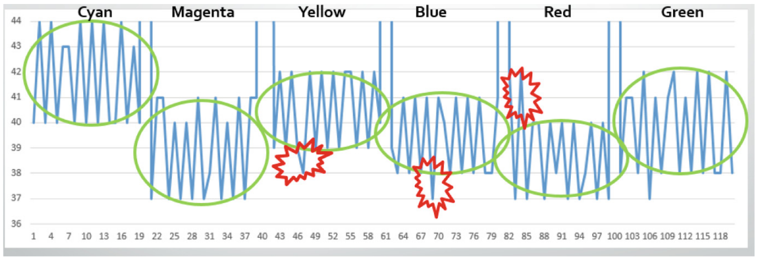


Fig. 9. Graph of data vs time (static) (Color figure online)

The range overlaps each other and the data differs from time to time due to environmental factor. Fortunately the pattern is consistent as the maximum and minimum range for each color is different. Thus, the color can be recognized by comparing with a default color, through determining with its maximum and minimum range value. This method requires a MR to be static for a short moment as moving MR unable to obtain a consistent range, as shown in Fig. 10.

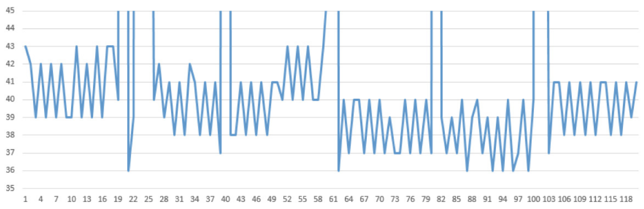


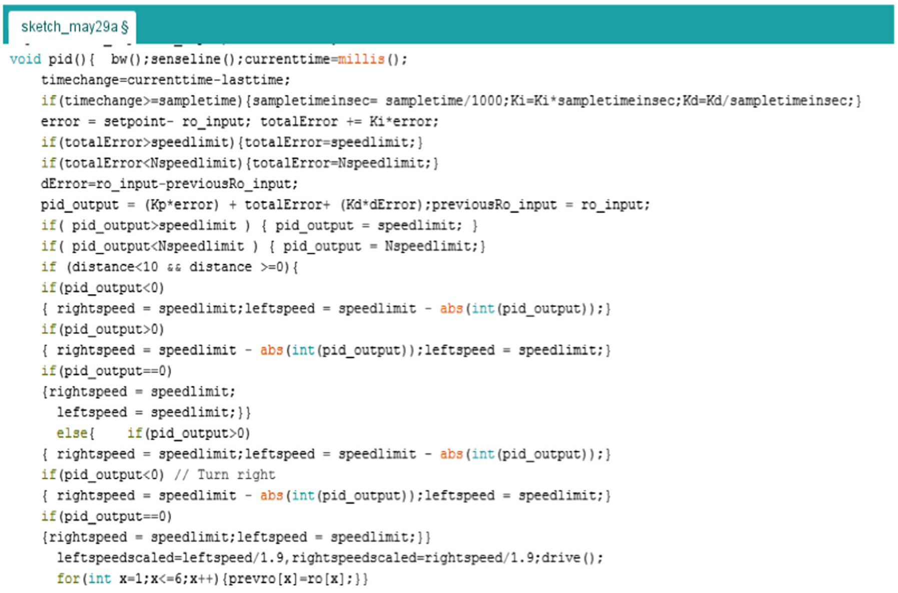
Fig. 10. Graph of data vs time (moving)

4 Conclusion

In this paper, a prototype of an autonomous guided MR based on microcontroller has been successfully fabricated. The feedback control was designed and tested able to follow the predefined path and perform path selection at junctions and reach designated location at a grid pattern field and best settling at $K_p = 88$, $K_i = 0.015$ while K_d have minor influences.

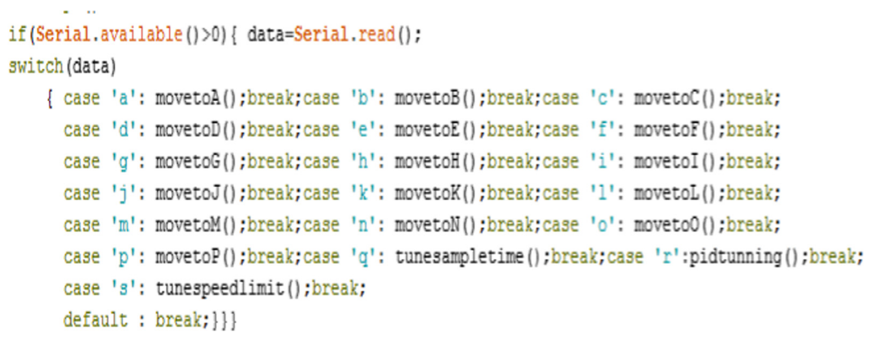
The prototype was able to achieve a 90° self-rotation will minimal skidding effect. The range of raw data from the reflective optical IR sensor for standard RGB and CYM color was also determined. Though, the MR also have some solvable handmade flaw issues. The stability of the prototype was tested and the result is acceptable. Still there are a lot of further improvement can be done to enhance the stability and functionality of the MR. The design of the MR can be further customized and studied for better performance. Some parts can be replaced as the first prototype contains defected parts that sometimes are issues during testing. Since Arduino UNO pins are fully utilized, a bigger capacity Arduino Mega is recommended to open up spaces for more added functions as the available pins increases.

Appendix 1


sketch_may29a §

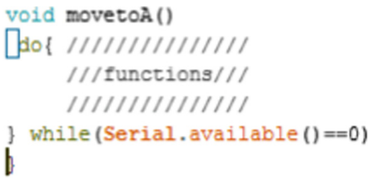
```
void pid(){  bw();senseline();currenttime=millis();
    timechange=currenttime-lasttime;
    if(timechange>=sampletime){sampletimeinsec= sampletime/1000;Ki=Ki*sampletimeinsec;Kd=Kd/sampletimeinsec;}
    error = setpoint- ro_input; totalError += Ki*error;
    if(totalError>speedlimit){totalError=speedlimit;}
    if(totalError<Nspeedlimit){totalError=Nspeedlimit;}
    dError=ro_input-previousRo_input;
    pid_output = (Kp*error) + totalError+ (Kd*dError);previousRo_input = ro_input;
    if( pid_output>speedlimit ) { pid_output = speedlimit; }
    if( pid_output<Nspeedlimit ) { pid_output = Nspeedlimit;}
    if (distance<10 && distance >=0){
    if(pid_output<0)
    { rightspeed = speedlimit;leftspeed = speedlimit - abs(int(pid_output));}
    if(pid_output>0)
    { rightspeed = speedlimit - abs(int(pid_output));leftspeed = speedlimit;}
    if(pid_output==0)
    {rightspeed = speedlimit;
      leftspeed = speedlimit;}}
      else{    if(pid_output>0)
    { rightspeed = speedlimit;leftspeed = speedlimit - abs(int(pid_output));}
    if(pid_output<0) // Turn right
    { rightspeed = speedlimit - abs(int(pid_output));leftspeed = speedlimit;}
    if(pid_output==0)
    {rightspeed = speedlimit;leftspeed = speedlimit;}}
      leftspeedscaled=leftspeed/1.9,rightspeedscaled=rightspeed/1.9;drive();
      for(int x=1;x<=6;x++){prevro[x]=ro[x];}}
```

Appendix 2: Serial Communication Program



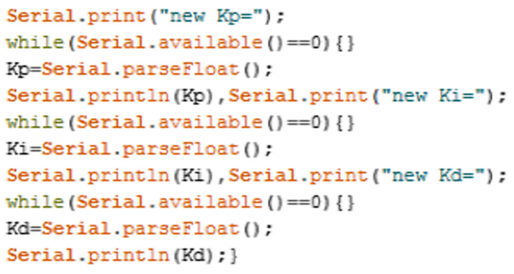
```
if(Serial.available()>0){ data=Serial.read();
switch(data)
    { case 'a': movetoA();break;case 'b': movetoB();break;case 'c': movetoC();break;
      case 'd': movetoD();break;case 'e': movetoE();break;case 'f': movetoF();break;
      case 'g': movetoG();break;case 'h': movetoH();break;case 'i': movetoI();break;
      case 'j': movetoJ();break;case 'k': movetoK();break;case 'l': movetoL();break;
      case 'm': movetoM();break;case 'n': movetoN();break;case 'o': movetoO();break;
      case 'p': movetoP();break;case 'q': tunesampletime();break;case 'r':pidtunning();break;
      case 's': tunespeedlimit();break;
      default : break;}}}
```

Appendix 3: Conditional Looping Program



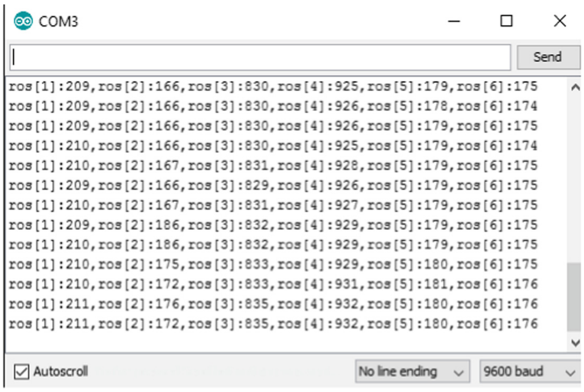
```
void movetoA()
{do{ ///////////////
     ///functions///
     ///////////////
} while(Serial.available()==0)
}
```

Appendix 4: Tuning Program



```
Serial.print("new Kp=");
while(Serial.available()==0){}
Kp=Serial.parseFloat();
Serial.println(Kp),Serial.print("new Ki=");
while(Serial.available()==0){}
Ki=Serial.parseFloat();
Serial.println(Ki),Serial.print("new Kd=");
while(Serial.available()==0){}
Kd=Serial.parseFloat();
Serial.println(Kd);}
```

Appendix 5: Serial Monitor Display of IR Sensor Raw Data



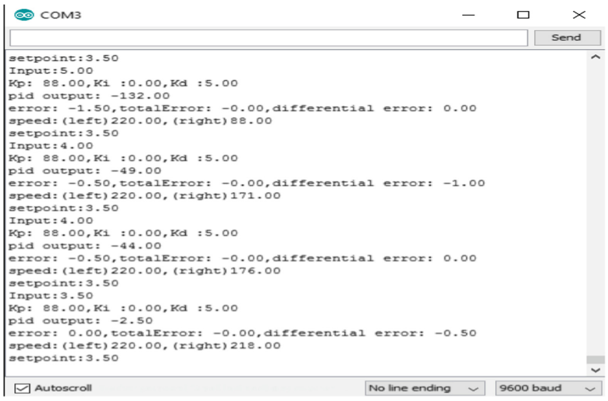
```
ros[1]:209,ros[2]:166,ros[3]:830,ros[4]:925,ros[5]:179,ros[6]:175
ros[1]:209,ros[2]:166,ros[3]:830,ros[4]:926,ros[5]:178,ros[6]:174
ros[1]:209,ros[2]:166,ros[3]:830,ros[4]:926,ros[5]:179,ros[6]:175
ros[1]:210,ros[2]:166,ros[3]:830,ros[4]:925,ros[5]:179,ros[6]:174
ros[1]:210,ros[2]:167,ros[3]:831,ros[4]:928,ros[5]:179,ros[6]:175
ros[1]:209,ros[2]:166,ros[3]:829,ros[4]:926,ros[5]:179,ros[6]:175
ros[1]:210,ros[2]:167,ros[3]:831,ros[4]:927,ros[5]:179,ros[6]:175
ros[1]:209,ros[2]:186,ros[3]:832,ros[4]:929,ros[5]:179,ros[6]:175
ros[1]:210,ros[2]:186,ros[3]:832,ros[4]:929,ros[5]:179,ros[6]:175
ros[1]:210,ros[2]:175,ros[3]:833,ros[4]:929,ros[5]:180,ros[6]:175
ros[1]:210,ros[2]:172,ros[3]:833,ros[4]:931,ros[5]:181,ros[6]:176
ros[1]:211,ros[2]:176,ros[3]:835,ros[4]:932,ros[5]:180,ros[6]:176
ros[1]:211,ros[2]:172,ros[3]:835,ros[4]:932,ros[5]:180,ros[6]:176
```

Autoscroll ✓ · No line ending · 9600 baud

Appendix 6: Comparator Program

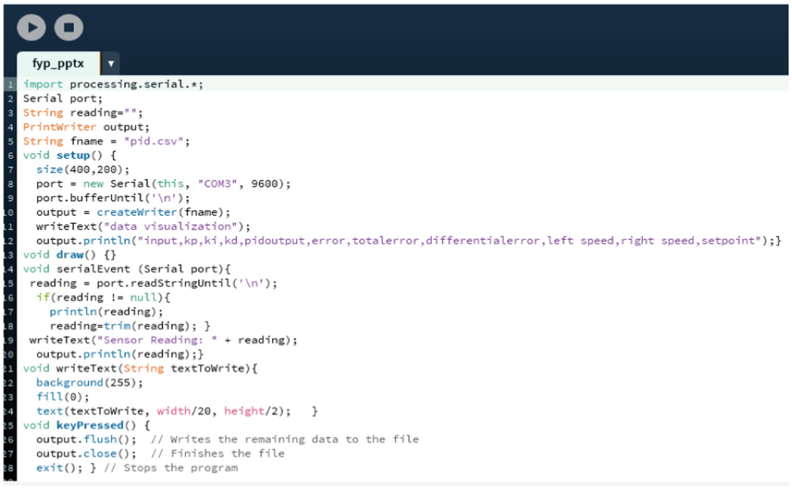
```
if (ros[1]>300)     { ro[1] = HIGH;}
           else                               { ro[1] = LOW;}
           if (ros[2]>300)     { ro[2] = HIGH;}
           else                               { ro[2] = LOW;}
           if (ros[3]>300)     { ro[3] = HIGH;}
           else                               { ro[3] = LOW;}
           if (ros[4]>300)     { ro[4] = HIGH;}
           else                               { ro[4] = LOW;}
           if (ros[5]>300)     { ro[5] = HIGH;}
           else                               { ro[5] = LOW;}
           if (ros[6]>300)     { ro[6] = HIGH;}
              else                               { ro[6] = LOW;}
```

Appendix 7: PID Feedback Parameter



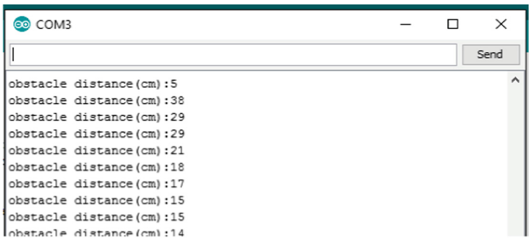
```
setpoint:3.50
Input:5.00
Kp: 88.00,Ki :0.00,Kd :5.00
pid output: -132.00
error: -1.50,totalError: -0.00,differential error: 0.00
speed:(left)220.00,(right)88.00
setpoint:3.50
Input:4.00
Kp: 88.00,Ki :0.00,Kd :5.00
pid output: -49.00
error: -0.50,totalError: -0.00,differential error: -1.00
speed:(left)220.00,(right)171.00
setpoint:3.50
Input:4.00
Kp: 88.00,Ki :0.00,Kd :5.00
pid output: -44.00
error: -0.50,totalError: -0.00,differential error: 0.00
speed:(left)220.00,(right)176.00
setpoint:3.50
Input:3.50
Kp: 88.00,Ki :0.00,Kd :5.00
pid output: -2.50
error: 0.00,totalError: -0.00,differential error: -0.50
speed:(left)220.00,(right)218.00
setpoint:3.50
```

Appendix 8: Processing Program



```
import processing.serial.*;
Serial port;
String reading="";
PrintWriter output;
String fname = "pid.csv";
void setup() {
  size(400,200);
  port = new Serial(this, "COM3", 9600);
  port.bufferUntil('\n');
  output = createWriter(fname);
  writeText("data visualization");
  output.println("input,kp,ki,kd,pidoutput,error,totalerror,differentialerror,left speed,right speed,setpoint");}
void draw() {}
void serialEvent (Serial port){
 reading = port.readStringUntil('\n');
  if(reading != null){
    println(reading);
    reading=trim(reading); }
 writeText("Sensor Reading: " + reading);
  output.println(reading);}
void writeText(String textToWrite){
  background(255);
  fill(0);
  text(textToWrite, width/20, height/2);   }
void keyPressed() {
  output.flush();  // Writes the remaining data to the file
  output.close();  // Finishes the file
  exit(); } // Stops the program
```

Appendix 9: Ultrasonic Processed Feedback



```
obstacle distance(cm):5
obstacle distance(cm):38
obstacle distance(cm):29
obstacle distance(cm):29
obstacle distance(cm):21
obstacle distance(cm):18
obstacle distance(cm):17
obstacle distance(cm):15
obstacle distance(cm):15
obstacle distance(cm):14
```

References

1. Spong, M.W., Vidyasagar, M.: Robot Dynamics and Control. Wiley, Hoboken (2008)
2. Hamza, M.F., Yap, H.J., Choudhury, I.A., Isa, A.I.: Application of kane's method for dynamic modeling of rotary inverted pendulum system
3. Tzafestas, S.G.: Introduction to Mobile Robot Control. Elsevier, Amsterdam (2013)
4. Hassan, Z.Z.: Automated guided vehicle (AGV) using 68HC11 microcontroller, Universiti Malaysia Pahang (2006)
5. Chapman, T.: Lab automation and robotics: automation on the move. Nature **421**, 661–666 (2003)
6. Royakkers, L., van Est, R.: A literature review on new robotics: automation from love to war. Int. J. Soc. Robot. **7**, 549–570 (2015)
7. Punetha, D., Kumar, N., Mehta, V.: Development and applications of line following robot based health care management system. Adv. Res. Comput. Eng. Technol. (IJARCET) **2**, 2446–2450 (2013)
8. Sumiya, T., Matsubara, Y., Nakano, M., Sugaya, M.: A mobile robot for fall detection for elderly-care. Procedia Comput. Sci. **60**, 870–880 (2015)
9. Singh, K., Singh, M., Gupta, N.: Design and implementation of cell-phone detection based line follower robot. Int. J. Electron. Comput. Sci. Eng.
10. Denei, S., Mastrogiovanni, F., Cannata, G.: Towards the creation of tactile maps for robots and their use in robot contact motion control. Robot. Auton. Syst. **63**, 293–308 (2015)
11. Normey-Rico, J.E., Alcala, I., Gómez-Ortega, J., Camacho, E.F.: Mobile robot path tracking using a robust PID controller. Control Eng. Pract. **9**, 1209–1214 (2001)
12. Durand, S., Marchand, N.: Further results on event-based PID controller. In: 2009 European Control Conference (ECC), pp. 1979–1984 (2009)
13. Smith, J., Campbell, S., Morton, J.: Design and implementation of a control algorithm for an autonomous lawnmower. In: 48th Midwest Symposium on Circuits and Systems, 2005, pp. 456–459 (2005)
14. Arduino, L.: Arduino Introduction, Arduino (2015). http://arduino.cc/en/guide/introduction
15. Schmidt, M.: Arduino. Pragmatic Bookshelf (2011)