Implementation of Particle Swarm Optimization for tuning of PID controller in Arduino Nano for Solar MPPT system

Muhammad Iqbal Mohd Zakki, Mohd Najib Mohd Hussain, and Nawawi Seroji

***Abstract*—This paper presents the implementation particle swarm optimization (PSO) embedded in Arduino Nano for proportional-integral-derivative controller (PID) tuning as a validation for real-time hardware application on the photovoltaic maximum power point tracking (MPPT) system. The Arduino is interfaced to the MATLAB/Simulink via serial communication device to provide duplex communication for PID tuning. The performance of PID tuning using PSO in Arduino was compared with the performance of MATLAB PID tuner toolbox based on simulation result via Simulink. The embedded PSO-PID provides better performance of 10 times higher compared to MATLAB PID tuner.**

***Index Terms*—PV, MPPT, PSO, PID, Arduino Nano.**

I. Introduction

PROPORTIONAL-INTEGRAL-DERIVATIVE (PID) controller are widely used in various industrial application for the controls of machines, and instruments. According to [1], the PID controller was first introduced in 1910. On some occasion, either the proportional P, proportional-integral PI, or proportional-derivative PD controllers is used instead of the PID, which depending on the application, and design. The controller is a mathematical based controller which is widely used in closed-loop system. They can offer control scheme such as reference input tracking, and disturbance rejection in the system's plant.

Application of PID controller in maximum power point tracking (MPPT) for photovoltaic (PV) system is to control the switching devices in the power converter. The absence of controllers such as lag-lead compensator, PID controller, or fuzzy logic controller in PV-MPPT system will causes the power optimization process for solar generator become slow, and prone to high steady-state error [2]–[6]. This will reduce the overall efficiency of PV solar energy harvesting.

Shown in Fig. 1 is the block diagram the control system. The control plant of the system, *G(S)*, includes PV module, DC-DC converter, and resistive load. The control input *R(s)* of the process loop is the voltage reference V_{ref} fed by MPPT controller, whereas the feedback signal *H(s)* is the instantaneous voltage of PV module V_{PV}. In this control scheme, the input of the PID controller is the error between V_{ref}, and V_{PV}.

There is numerous method of PID tuning. Some of them are by using the Ziegler-Nichols' (Z-N) technique, the Cohen-Coon technique, and gain-phase margin technique [7]. Nevertheless, the PID controller can be also tuned using computation intelligence (CI) techniques by either self-adaptive fuzzy, or evolutionary computation (EC) [8].

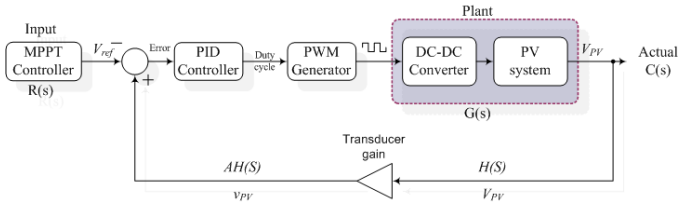

Fig. 1. General block diagram for the closed-loop PV-MPPT system

The PID tuning method proposed in this paper is by using the particle swarm optimization (PSO) algorithm [9] due to the fact that this method independent from human expertise in tuning process, while can be easily applied to different PV system [10]. Furthermore, PSO algorithm is preferred for PID optimization process instead other algorithms in evolutionary algorithm (EA) family because the PSO have only three steps in the algorithm [11]. Due to low requirement of memory capacity for data processing, thus, PSO is more viable for the implementation on low-cost real-time hardware.

II. Photovoltaic MPPT System

PV system is an electrical generation system that can be built from integration of PV modules; that make-up from PV cells. PV cells are made of semiconductor materials, with similar working principles to other semiconductor devices such as photo-transistor, photo-resistors, photo-detectors, and others. As a result, PV cells also inherit the non-linear electrical current-to-voltage (*I-V)* relationship that are rely on the ambient environment such as the intensity of incidence light (solar irradiance ψ), and the temperature of the solar cell T_{cell}.

These two factors produces different *I-V* curve and their

Submitted on 28th of February 2018 and accepted on 28th August 2019. The author would like to thank Universiti Teknologi Mara Malaysia and Research Management Institute for the financial support through Fundamental Research Grant Scheme (FRGS/1/2016/TK10/UiTM/62/2) for this study.

Muhammad Iqbal Mohd Zakki is with Universiti Teknologi Mara Pulau Pinang, Permatang Pauh, Jalan Permatang Pauh, 13500 Permatang Pauh, Pulau Pinang, Malaysia (e-mail: muhammadiqbal.mohdzakki@live.com).

Dr. Mohd Najib Mohd Hussain is with Universiti Teknologi Mara Pulau Pinang, Permatang Pauh, Jalan Permatang Pauh, 13500 Permatang Pauh, Pulau Pinang, Malaysia.

corresponding power-to-voltage (P-V) curve as shown in Fig. 2. A PV system that directly connected to the electrical load will operate at non-optimum parameters because the operating points of PV cells are impedance-dependent. Moreover, the optimum operating parameters, or the maximum power point (MPP) changes dynamically with insolation, and heat acts on the solar cell. Hence, the PV system requires devices to optimize the operating parameters to the MPP by employs MPPT system.

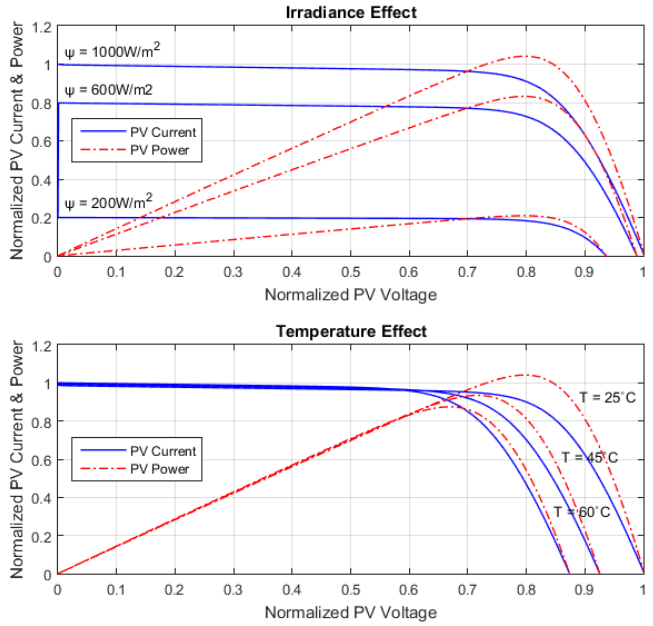

Fig. 2. Operating environment effect on PV cell

Generally, MPPT system consist of intermediary power converter, MPPT controller, and control loop as exemplified in Fig. 3. The DC-DC chopper (buck-boost) is controlled by PID controller, based on the command from MPPT. In this case, the power converter is controlled by PID controller, which receives inputs from MPPT reference voltage, V_{ref}, and instantaneous PV voltage, V_{PV}.

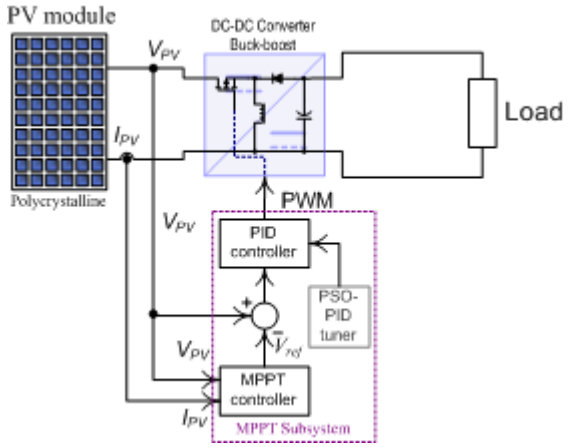

Fig. 3. Detailed closed loop control of PV-MPPT system

The prime focus of this paper is to verify the feasibility of PSO implementation into low-cost microcontroller hardware for real-time application by comparing the performance of the external hardware-embedded PSO-PID tuning with the internal MATLAB PID tuner. Both tuning method are applied on the PV-MPPT system model in the Simulink. The performance of these tuning method are compared in terms of accumulative mean error squared (MSE).

III. PID Tuning Methods

It is worth to mention that Simulink is used as an intrinsic control plant for performance analysis of the PID tuning as a method to eliminate influence of external disturbances as founded in the real hardware environment. This evaluation approach is carried out so that the C++ coding of embedded PSO-PID algorithm can be tested, debugged, and verified. On the other hand, a fair comparison of performance with the MATLAB PID tuner can be made via Simulink environment.

Shown in Fig. 4 is comprehensive Simulink blocks of PV system comprise of PV module block model, MPPT, MPPT control system, voltage and current sensors, and the buck-boost converter. The system plant consist of PV panel, MPPT controlled DC-DC converter, and load.

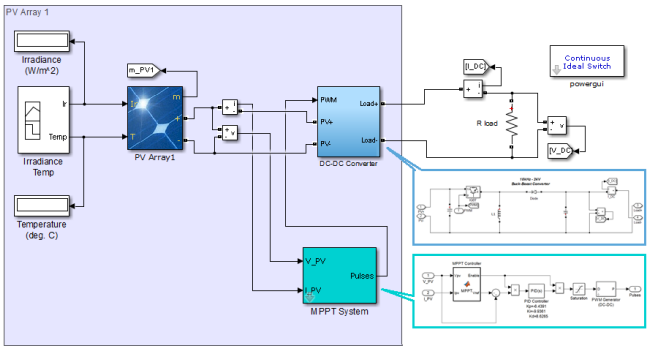

Fig. 4. Simulink PV-MPPT system

Fig. 5 shows the internal components of MPPT subsystem. According to Fig. 5, the MPPT controller receives input measurements of V_{PV}, and I_{PV}, while process the tracking reference V_{ref} as command input to the control loop. PID controller receives error signal from the comparison of V_{ref} and V_{PV} on the summing junction. The MPPT also controls the input and output of PID controller via 'Enable' port.

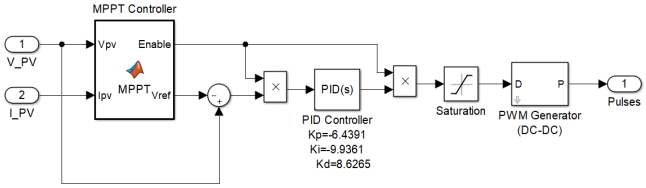

Fig. 5. MPPT controller with PID closed-loop

Fig. 6 displays the MPPT inverted buck-boost DC-DC chopper driven by the control loop as presented in Fig. 5.

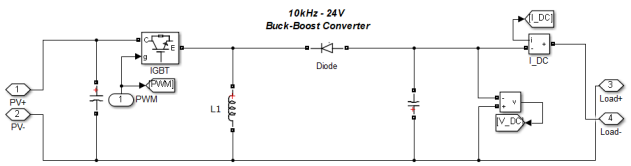

Fig. 6. DC-DC buck-boost converter

A. Arduino PSO-PID Tuning

PSO-PID tuning via Arduino Nano hardware is carried out by interconnecting the MATLAB/Simulink with Arduino board by Serial-to USB communication as illustrated in Fig. 7. The PSO-PID tuning algorithm was embedded into the microcontroller to provide PID gain parameter, while the MATLAB/Simulink accomplishes the simulation process to obtain the fitness for each particles position.

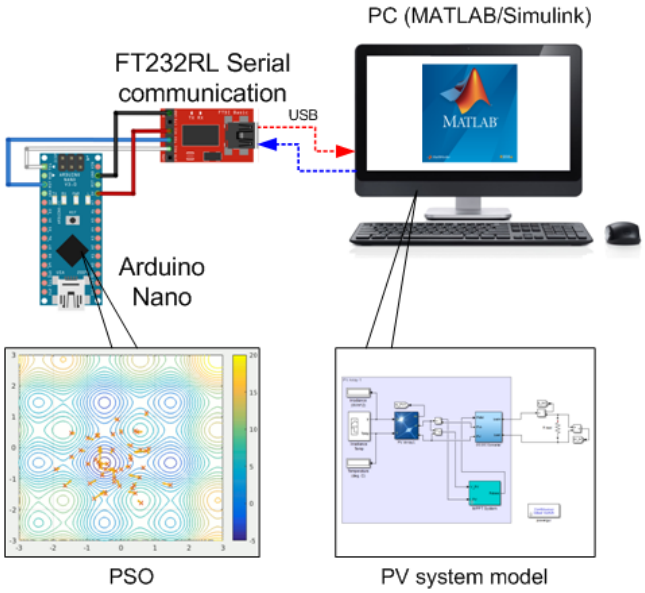

Fig. 7. Block diagram illustration for Arduino PSO-PID tuning

1) The Particle Swarm Optimization (PSO)

The particle swarm optimization algorithm was introduced in 1995 by Kennedy, and Eberhart [9]. The study presents a simple algorithm derived from the social interaction concept of birds' flock that can be used to find solutions for wide range of non-linear problems by using metaheuristics search technique. Conceptually, it has the similarities with genetic algorithm (GA) since it uses the notation of "fitness" like other family members of evolutionary algorithm (EA) in computation intelligence (CI) to represent the performance of a solution from the population.

According to [11], PSO uses less effort in computation to achieve the same high quality solutions compared to GA. The study present that PSO is proven its advantage with 99% statistical confidence. Therefore, the PSO was likely more efficient to be employed into low-cost hardware systems for online system application such as PV-MPPT system as this study. For that reason, PSO was implemented as PID tuner in this study.

Fundamentally, PSO algorithm searches the optimum parameters by iteratively updates the particles' velocity and position until the fitness, and objective of the optimization are met.

According to Fig. 8, after the initialization, the algorithm waits for iterative fitness computation from individual particle position. Subsequently, the current best particle position p_{best} is defined by determine the lowest MSE in the swarm fitness.

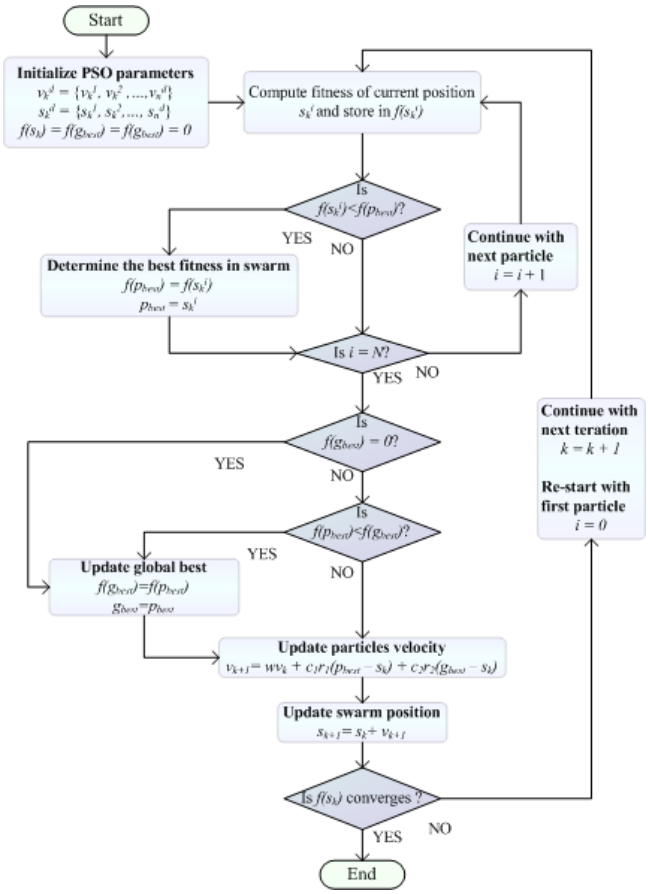

Fig. 8. PSO-PID algorithm flowchart

After the fitness of all particle in the population have been testified individually, the current best fitness $f(p_{best})$ of the swarm is compared to the global best fitness $f(g_{best})$. The fitness functions for each particle were evaluated as mean error squared, MSE of input voltage reference, V_{ref} and the actual instantaneous voltage, V_{PV}. The fitness function for PSO is shown in (1).

$$f(s_i^k) = \frac{1}{T}\sum_{i=1}^{T}(V_{ref}(t) - V_{PV}(t))^2 \quad (1)$$

where $f(s_i^k)$ is the fitness of the current particle, and T is the total simulation data point.

The comparison to determine the p_{best}, and g_{best} depends on the "objective function" of the optimization process. In the case of PID tuning, it is necessary for the control system to have the lowest error as possible, thus, the resultant "objective function" is to determine p_{best}, and g_{best} of the lowest MSE as shown in Fig. 8.

Essentially, the PSO uses equation (2) and (3) as the respective velocity and position update function for its particle movement. The influence of the swarm and the global optima position on the particle's position is illustrated as in Fig. 9.

$$v_i^{k+1} = wv_i^k + c_1 r_1 (p_{best} - s_i^k) + c_2 r_2 (g_{best} - s_i^k) \quad (2)$$

$$s_i^{k+1} = s_i^k + v_i^{k+1} \quad (3)$$

where,

k is the k[th] iteration;

i is the particle number of a given swarm;

r_1 and r_2 is random number as learning factor for individual particle to prevent entrapment of particles in local optima

v_i^k is the current particle velocity;

s_i^k is the current particle position;

w is the swarm weight, or inertia factor;

c_1, and c_2 are the particles, and swarm confidence factor;

p_{best} is particle position with the best fitness in the swarm, or iteration;

g_{best} is the global best particle position.

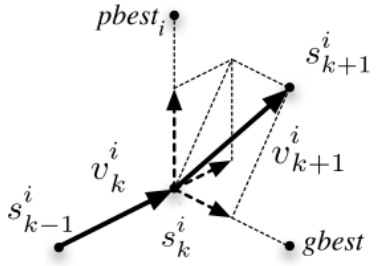

Fig. 9. Movement of particle under swarm influence

2) Hardware Implementations

The PSO-PID tuning was carried out using hardware as shown in Fig. 10. This tuning procedure employs master/slave programming; hardware embedded program (master), and MATLAB m-file (slave). The PSO-PID tuning program is embedded in microcontroller, while the m-file program in MATLAB is used to parse parameters between hardware, and Simulink, back and forth. The device for hardware-to-software interfacing is shown in Fig. 11.

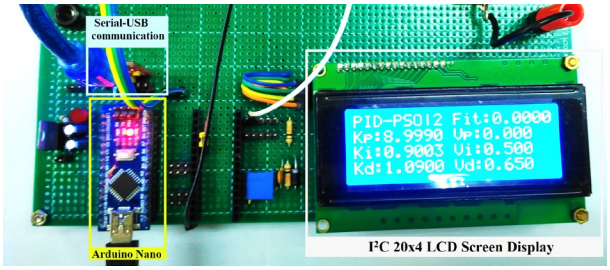

Fig. 10. Hardware for PSO-PID tuning on Simulink model

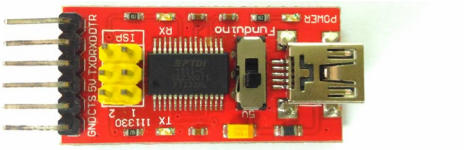

Fig. 11. USB-Serial FTDI FT232RL communication chip breakout board.

Fig. 12 shows the program flowchart for PSO-PID tuning via hardware-to-MATLAB interface. In the beginning, the PSO parameters initialize in the hardware memory, and wait for MATLAB slave acknowledgement before begin transmitting the PID parameter. The PID gains from the Arduino will be parsed to Simulink. Subsequently, the MSE fitness data from the simulation run will be sent back to the master hardware. The optimization process carried out by the master device iteratively until the fitness of the population converges.

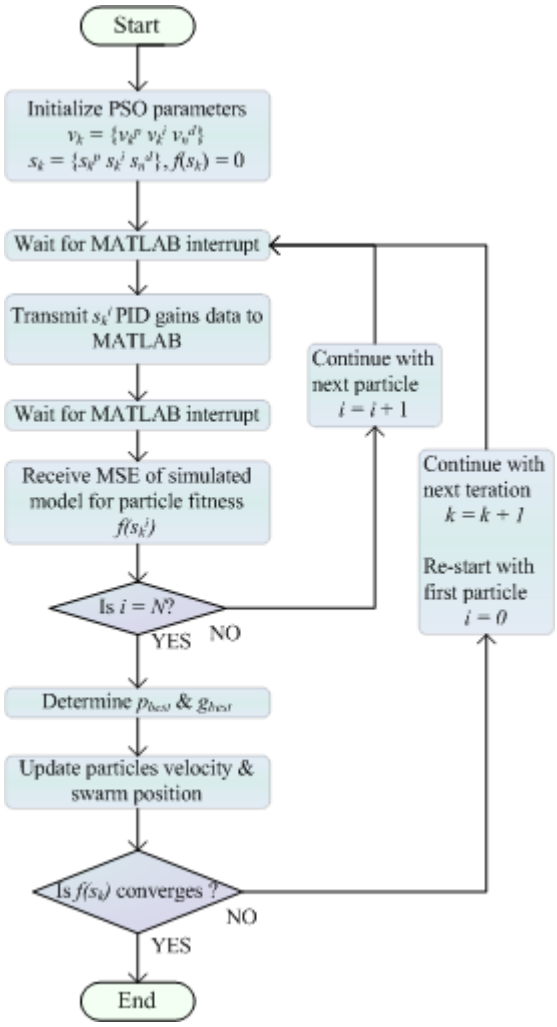

Fig. 12. Flowchart of embedded PSO-PID program (master)

Shown in Fig. 13 is the output verbose of the code compiling for PSO-PID MATLAB tuner embedded on Arduino Nano. The memory consumption for the program and variables are 47% of 30720 byte, and 55% of 2048 byte, respectively.

```
Archiving built core (caching) in: C:\Users\muham_000\AppData\Loca
Linking everything together...
"C:\Program Files (x86)\Arduino\hardware\tools\avr/bin/avr-gcc" -w
"C:\Program Files (x86)\Arduino\hardware\tools\avr/bin/avr-objcopy
"C:\Program Files (x86)\Arduino\hardware\tools\avr/bin/avr-objcopy
Using library NewliquidCrystal in folder: C:\Users\muham_000\Docum
Using library Wire at version 1.0 in folder: C:\Program Files (x86
Using library SoftwareSerial at version 1.0 in folder: C:\Program
Sketch uses 14744 bytes (47%) of program storage space. Maximum is
Global variables use 1139 bytes (55%) of dynamic memory, leaving 9
```

Fig. 13. PSO-PID MATLAB tuner code compilation output verbose

Fig. 14 and Fig. 15 shows snippet of PSO-PID coding for velocity, and position update function.

```
for (i = 0; i < num; i++)
{
  vKp[i] = vKp[i] * w + (c1 * r1 * (pbest_Kp - Kp[i])) + (c2 * r2 * (gbest_Kp - Kp[i]))
  vKi[i] = vKi[i] * w + (c1 * r1 * (pbest_Ki - Ki[i])) + (c2 * r2 * (gbest_Ki - Ki[i]))
  vKd[i] = vKd[i] * w + (c1 * r1 * (pbest_Kd - Kd[i])) + (c2 * r2 * (gbest_Kd - Kd[i]))
  if (vKp[i] > velocity_limit)vKp[i] = velocity_limit;
  else if (vKp[i] < -velocity_limit)vKp[i] = -velocity_limit;
```

Fig. 14. PSO velocity update function

```
  Kp[i] = Kp[i] + vKp[i];                                    //Position update for K
  Ki[i] = Ki[i] + vKi[i];                                    //Position update for K
  Kd[i] = Kd[i] + vKd[i];                                    //Position update for K
  if (Kp[i] > position_limit)Kp[i] = position_limit;
  else if (Kp[i] < -position_limit)Kp[i] = -position_limit;
```

Fig. 15. PSO position update function

The Fig. 16 shows the running process of PSO-PID tuning in the MATLAB command window, while Fig. 17 shows the simulation run to find the particle fitness. The fitness of the particle is determined by the mean sum of squared error between the reference signal V_{ref} and the instantaneous PV voltage, V_{PV}.

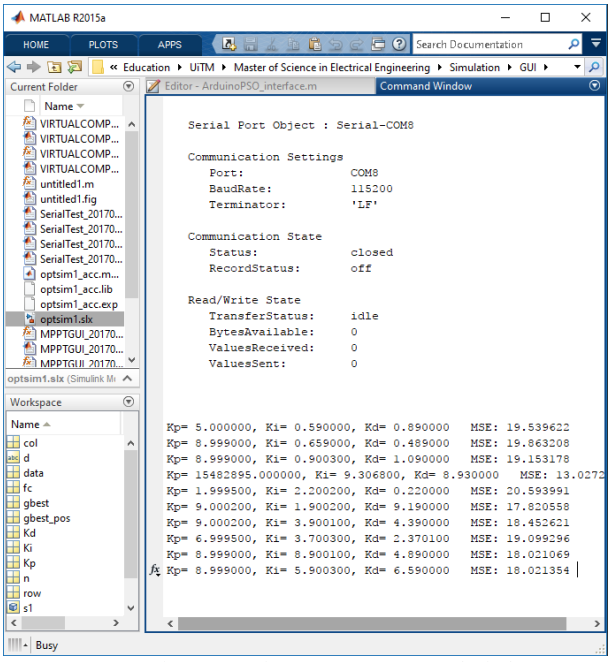

Fig. 16. PSO-PID tuning process in MATLAB command window

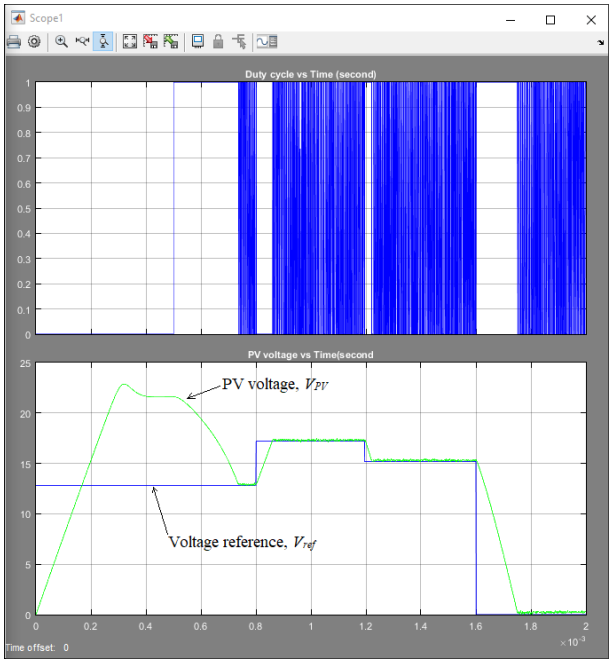

Fig. 17. Simulation runs for PSO-PID tuning

The iteration of the PSO stopped after the fitness between particles in the swarm converges with the standard deviation lower than 0.1. The optimized PID parameters were returned by the g_{best} position as follows;

- Proportional gain, $K_p = 6.4391$
- Integral gain, $K_i = 9.9361$
- Derivative gain, $K_d = 8.6265$
- Derivative filter, $N = 100$

B. MATLAB PID Tuner Toolbox

Since the PV system consist of non-linear system response due to the PV module, and non-linear switching response of the power converters, the plant of the system must be linearized by simplify the plant with mathematical modelling via system identification process.

As shown in Fig. 18, the process initiates from input-output (I/O) response data sampling. Subsequently, the plant is estimated and the fitness of the model output is compared with the sampled data. If the fitness of the model is less than 80% of the I/O sample, the plant will be re-modelled with new structure. Otherwise, the linearized model will be applied into PID tuner toolbox for tuning. Nonetheless, this linearized model only used in the PID tuner toolbox to simplify the non-linearity of the system plant.

The performance of the newly acquired PID parameters will be tested with the simulation blocks as demonstrated in Fig. 4, by replacing the MPPT with a step input, purposely to compare with the performance PSO tuned PID controller.

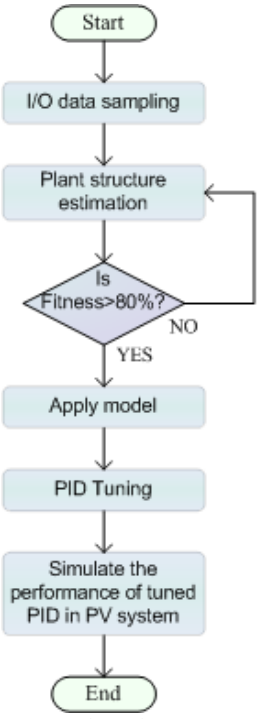

Fig. 18. Process flows for PID tuning using MATLAB PID tuner toolbox

1) Input-Output Sampling

To begin with, the I/O data of the system was sampled by repeatedly inject the step input directly into the input of the PID controller, while bypassing the input from MPPT and the feedback loop. Thus, the control system operates as open-loop control. The PID controller is set to P-only mode with proportional gain, $K_p = 1$ for linear modelling purpose.

Fig. 19 shows present I/O sampling result using step input with rise time at 0.0005s, until 0.01s. The sample time T_s for I/O data sample is $1x10^{-6}$s per iteration.

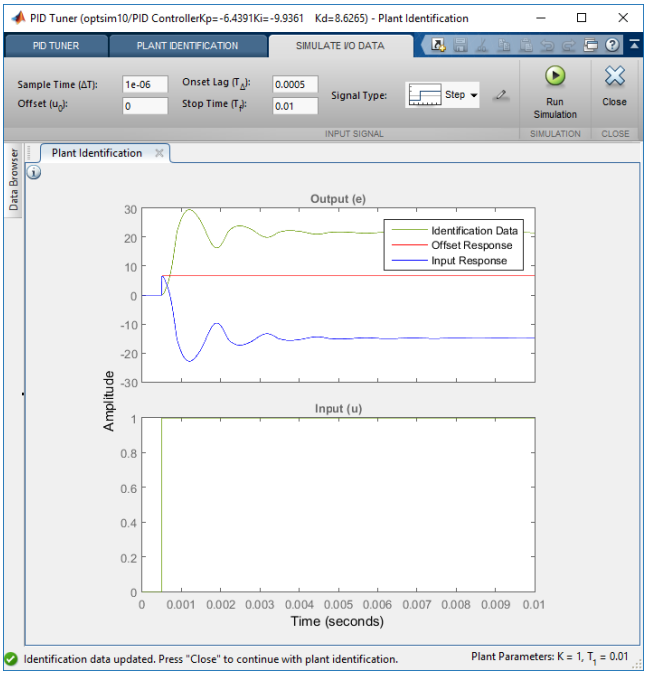

Fig. 19. I/O response data sampling for plant identification

2) Plant Identification

After the data sample is obtained, the model is then estimated via system identification approach by using structures by either of one-pole, two real-poles, underdamped pair, and underdamped with real-pole transfer function, and state-space. All of the mentioned structures are tested while their fitness was compared to each other. In conclusion, underdamped pair have the highest fitness value besides having the simpler form of equation compared to state-space. Exhibited in Fig. 20 is the linear plant modelling using underdamped pair transfer function. The angular frequency, T_ω, and damping ratio, ζ are manually adjusted so that the identified data intersect as close as possible on the identification data obtained from the I/O sampling.

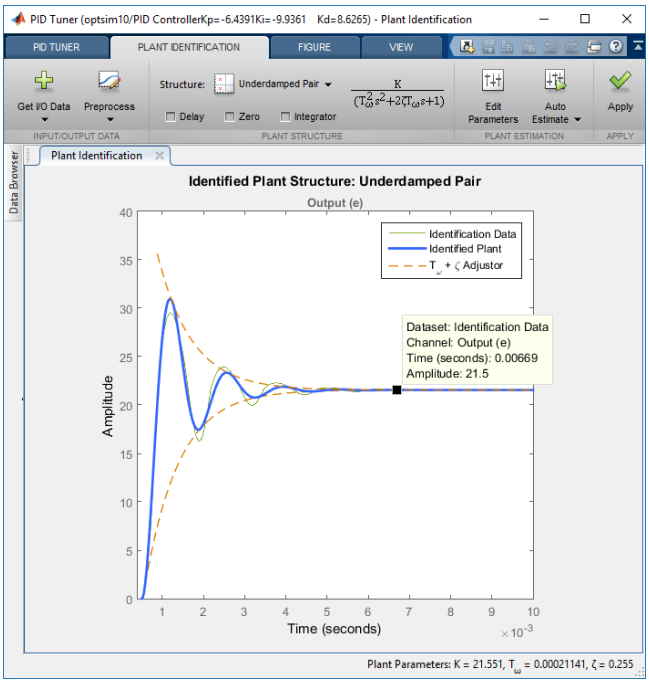

Fig. 20. Linear model plant estimation

The identified model is further fined-tuned by using the auto estimate function of the toolbox in order to achieved fitness of 87.37% as exemplified in Fig. 21.

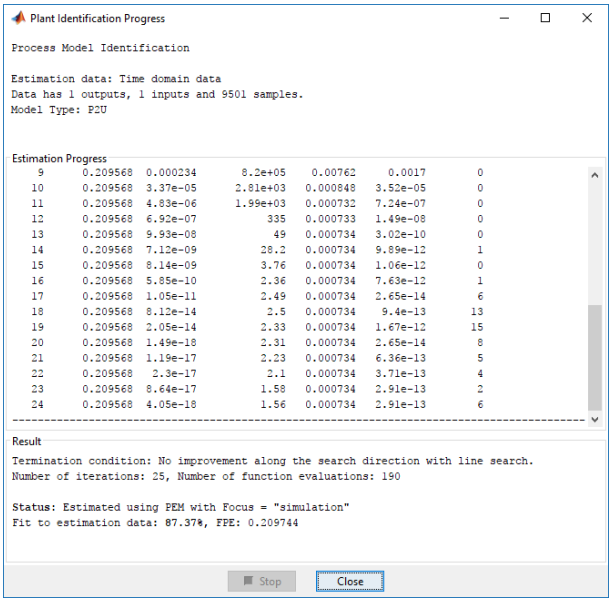

Fig. 21. Plant identification for linear model

The yield of the modelling process for the underdamped pair transfer function (4) as below;

$$G(s) = \frac{K}{T_\omega{}^2 s^2 + 2\zeta T_\omega s + 1} \tag{4}$$

where the value of gain, K, angular frequency, T_ω, and damping ratio, ζ are 21.551, 0.00021141, and 0.255 respectively. Thus, the transfer function for linear model is simplified as in (5).

$$G(s) = \frac{21.551}{0.000000044694 s^2 + 0.00010782 s + 1} \tag{5}$$

3) PID Tuning

Fig. 22 and Fig. 23 are the PID tuner window, and PID controller specifications, correspondingly. The tuner can simply be used by adjust the control slider for the response time, and the transient behaviours. It is desirable that the controller to be able to provide the response time, oscillations, and errors as minimal as possible. From the tuned parameters, the achieved rise time is 0.00002s, with 0% overshoot.

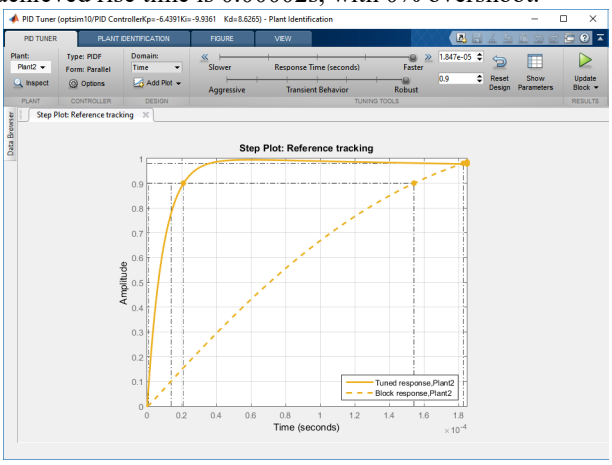

Fig. 22. MATLAB PID tuner tool

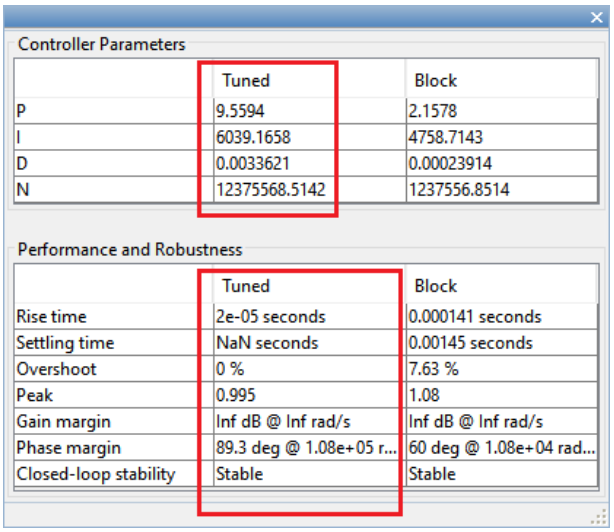

Fig. 23. MATLAB PID parameter and performance

It is worth noting that, the PID tuning by the toolbox uses the linearized plant model, therefore, the performance might be different when the tuned PID parameter applied into the non-linear PV system.

IV. Performance Comparison

By utilizing both methods to obtain PID gain parameters, the performance of the controller is evaluated in simulation stage from Simulink block as shown in the Fig. 4. The system block is injected with unit step input with amplitude of 15V reference voltage, V_{ref}. The parameters for both tuning method are assessed in Table 1. Fig. 24 and Fig. 25 show the step response of closed loop PID control response using MATLAB PID tuner toolbox, and Arduino PSO-PID tuning, respectively.

TABLE I
SIMULATION PARAMETERS OF PID TUNING METHOD

PID tuning method	MATLAB PID Tuner Toolbox	Arduino PSO-PID Tuner
Proportional gain, *Kp*	9.5594	6.4391
Integral gain, *Ki*	6039.1658	9.9361
Derivative gain, *Kd*	0.003621	8.6265
Derivative filter, *N*	12375568.5142	100
Settling time (seconds)	0.003501	0.000203
Mean squared error, MSE	1.009891	0.1025

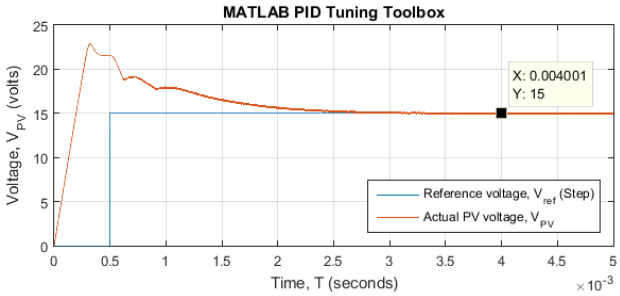

Fig. 24. Step response of closed loop PID control by MATLAB PID Tuner

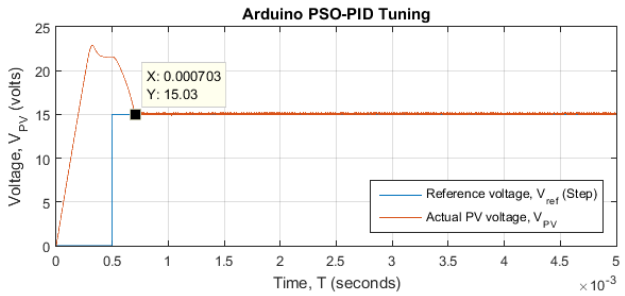

Fig. 25. Step response of closed loop PID control by PSO-PID tuning

From the step response test, it is observed that PSO-PID method has the fastest response time compared to the MATLAB-tuned controller. The PSO-PID method achieves steady-state at t = 0.703ms. Thus, the time taken for the plant output to settle to step input t_{ss} is 0.203ms (note that unit step rise time t_r = 0.5ms). In contrast, MATLAB-tuned PID takes time to settle for 3.501ms. The recorded MSE for MATLAB-tuned PID is 1.009891. While for PSO-PID method, MSE successively reduced to 0.1025.

Since the embedded PSO-PID tuning only takes about one-tenth of the response time of MATLAB-tuner method, therefore, the PSO-PID shows its advantage over the counterpart.

V. Conclusion

From the following results, the PSO-PID algorithm embedded in the 8–bit microcontroller proves its feasibility, and effectiveness by successfully optimize the PID controller by provide about 10 times improvement compared to MATLAB PID tuner in terms of MSE, and response time. Moreover, PSO-PID requires less step in tuning procedure since it does not require collection of data for off-line system identification. Besides that, the tuning process is also reliable and independent from human supervision. Therefore, the PSO-PID can be identified as an adaptive tuning method that can directly embedded to the real PV-MPPT hardware of any setup. Furthermore, the proposed PSO-PID tuning method consumes low capacity of flash memory, and processing power. Hence, PSO is viable for the implementation in low-cost microcontroller hardware.

VI. References

[1] M. Amirinejad, M. Eslami, and A. Noori, "Automatic PID Controller Parameter Tuning Using Bees Algorithm," vol. 5, no. 8, pp. 24–28, 2014.

[2] N. Femia, G. Petrone, G. Spagnuolo, and M. Vitelli, *Power electronics and control techniques for maximum energy harvesting in photovoltaic systems*. 2013.

[3] M. H. (Muhammad H. Rashid, *Power electronics handbook : devices, circuits, and applications*. Butterworth-Heinemann, 2011.

[4] Y. Wang, L. Ding, and N. Li, "The application of fuzzy parameters self-tuning PID controller in MPPT of photovoltaic power system," *Proc. 2011 Int. Conf. Transp. Mech. Electr. Eng. TMEE 2011*, no. Figure 2, pp. 1129–1132, 2011.

[5] A. El Khateb, N. A. Rahim, and J. Selvaraj, "Optimized PID controller for both single phase inverter and MPPT SEPIC DC/DC converter of PV module," *2011 IEEE Int. Electr. Mach. Drives Conf. IEMDC 2011*, pp. 1036–1041, 2011.

[6] R. Pradhan and B. Subudhi, "Design and real-time implementation of a new auto-tuned adaptive MPPT control for a photovoltaic system," *Int. J. Electr. Power Energy Syst.*, vol. 64, pp. 792–803, 2015.

[7] A. Bagis, "Determination of the PID controller parameters by modified genetic algorithm for improved performance," *J. Inf. Sci. Eng.*, vol. 23, no. 5, pp. 1469–1480, 2007.

[8] A. A. R. Coelho, "PID techniques in intelligent and adaptive algorithms," in *38th Midwest Symposium on Circuits and Systems. Proceedings*, 1996, vol. 1, pp. 409–412.

[9] J. Kennedy and R. Eberhart, "Particle Swarm Optimization," pp. 1942–1948, 1995.

[10] G. Reynoso-Meza, J. Sanchis, J. M. Herrero, and C. Ramos, "Evolutionary auto-tuning algorithm for PID controllers," *IFAC Proc. Vol.*, vol. 2, no. PART 1, pp. 631–636, 2012.

[11] R. Hassan and B. Cohanim, "A comparison of particle swarm optimization and the genetic algorithm," *1st AIAA Multidiscip. Des. Optim. Spec. Conf.*, pp. 1–13, 2005.

Muhammad Iqbal, Mohd Zakki received Diploma in Electrical Engineering (Power) from Universiti Teknologi Mara Pulau Pinang, in 2012, and B.Sc in Engineering (Electrical and Electronics) from the same university, in 2017. He is currently pursuing the M.Sc in Electrical Engineering by research in the field of renewable energy, specifically in optimization of solar photovoltaics system. His research interest include development of PV-MPPT system, embedded electronics, power electronics and energy conversion, artificial intelligence, and system modelling.

Currently, he works as research assistant in the university. In 2015, he used to work as an intern at Sungai Petani Town Municipal Council (Majlis Perbandaran Sungai Petani Kedah, MPSPK) in Engineering Department.

Mr. Muhammad Iqbal is a graduate member of Board of Engineers Malaysia (BEM), and the Institutions of Engineers Malaysia (IEM).