HYBRID POWER SYSTEM DESIGN FOR AUTONOMOUS GROUND ROBOTS

by

Ali Qahtan Al-Tameemi

A Thesis Presented to the Faculty of the
American University of Sharjah
College of Engineering
in Partial Fulfillment
of the Requirements
for the Degree of

Master of Science in
Mechatronics Engineering

Sharjah, United Arab Emirates

November 2018

Approval Signatures

We, the undersigned, approve the Master's Thesis of Ali Qahtan Al-Tameemi

Thesis Title: Hybrid Power System Design for Autonomous Ground Robots

Signature	**Date of Signature** (dd/mm/yyyy)
_____	_____
Dr. Shayok Mukhopadhyay Assistant Professor, Department of Electrical Engineering Thesis Advisor	
_____	_____
Dr. Lotfi Romdhane Professor, Department of Mechanical Engineering Thesis Committee Member	
_____	_____
Dr. Usman Tariq Assistant Professor, Department of Electrical Engineering Thesis Committee Member	
_____	_____
Dr. Lotfi Romdhane Director, Mechatronics Engineering Graduate Program	
_____	_____
Dr. Ghaleb Husseini Associate Dean for Graduate Affairs and Research College of Engineering	
_____	_____
Dr. Richard Schoephoerster Dean, College of Engineering	
_____	_____
Dr.Mohamed El-Tarhuni Vice Provost for Graduate Studies	

Acknowledgement

I would like to thank my advisor Dr. Shayok Mukhopadhyay for his support, exceptional assistance, novel ideas, and worthy discussions. I would also like to thank the professors involved with Mechatronics Engineering graduate program for providing knowledge. In addition, I would like to thank Mr. Wassim Al-Masri and other friends for their extraordinary effort. Finally, I would like to thank the American University of Sharjah for the chance to be a Graduate Teaching Assistant, as it considerably helped me in this work.

Dedication

Dedicated to my family...

Abstract

The interest in mobile robots has increased rapidly due to the complicated tasks a mobile robot can accomplish. An efficient robot power supply system can increase the robot range of travel. Different power management techniques have been applied heavily in the field of electric vehicles. Such techniques are helpful in terms of extending the robot driving range; power controller requires placing a DC converter that consists of power switches, inductors, and capacitors. In most cases, robots are still powered by a single battery. This observation inspired this work to develop an enhanced passive multi-source power system, using Generalized Predictive Control (GPC) and Kalman filtering (KF) to find the minimum power required to drive the robot along a predefined path. As a result, the designed power system extends robot driving range from 3 to 16 hours. Since batteries are a major component of any current hybrid energy system design, any good energy management system must incorporate an impending battery failure detection system, so that other energy sources can be switched on to replace a dying battery. This work proposes a battery voltage collapse detection technique based on Fast Fourier Transforms (FFTs) and artificial neural network (ANNs), where the robot driving range is extended more by 3 hours using a backup battery. This work aims to use two batteries, a supercapacitor, and a fuel cell based system to form a long-lasting hybrid energy system for a mobile robot.

Search Terms: ***Generalized Predictive Control (GPC), Kalman filter (KF), Fast Fourier Transform (FFT), and Artificial Neural Network (ANN)***

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction and Literature Review

1.1. Introduction

Power management strategies are applied heavily in the field of electric vehicles to manage more than one power source, so that battery lifetime can be extended, and the driving range of the electric vehicle can be increased. Recently, mobile robots have become an effective tool in navigation, exploration, firefighting, and other applications where there may be an inherent danger to humans. Utilizing hybrid energy sources on board a robot, and developing a power management strategy, can increase the driving range of mobile robots. This is very important, because if a robot suffers a malfunction in any system, an extended driving range can improve the chances of the robot navigating home. This can reduce the number of man-hours required to conduct a salvage operation for the robot, and maybe reduce the number of losses of expensive scientific equipment far away from a robot's operating home base. A preliminary example of this is in [1], where NASA successfully deployed two mobile robots on Mars for environmental exploration purposes. The two robots were powered by Lithium-ion batteries combined with photovoltaic cells. As a result, the robots could last in operation for three months before power sources were completely exhausted. Based on the above, this work aims to use two batteries, a supercapacitor, and a fuel cell based system to form a long-lasting hybrid energy system for a mobile robot.

1.2. Literature Review

Normally, mobile robots are powered by a single source. As discussed in [2], a mobile robot was converted from a lead acid battery to a hydrogen PEM fuel cell. As a result, the robot's performance was influenced due to additional weight. However, the recharging time, compared to battery operated robots, was eliminated due to the requirement of a single refueling step. Another example is illustrated in [3]; a battery connected in parallel with a supercapacitor extends battery lifetime due to reducing battery discharging current; however, the proposed parallel combination reduced vehicle performance because of additional weight.

1.2.1. Power sources configurations. Several energy system topologies have been discussed in the literature [4], [5], [6], and [7]; where the most basic power sources combination is seen in Fig. 1. This type of configuration is called passive parallel connection, and it is the simplest way to connect different power sources to each other. The simplicity of electric circuit implementation and low cost are the main advantages behind applying such configuration. On the other hand, the major drawback is the absence of any type of control on power sources. As mentioned in [4] and [5], the passive parallel connection can be improved more by using DC/DC converters in order to control one of the power sources separately as seen for example in Fig. 2 (a) and (b).

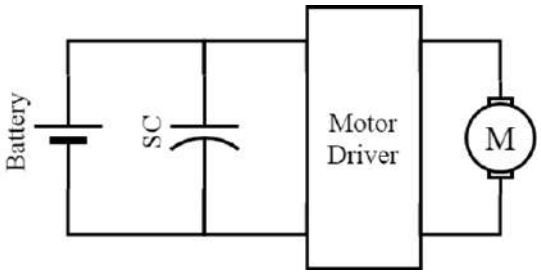

Figure 1: Passive configuration

The semi-active configuration seen in Fig. 2 adds more flexibly to the system by decoupling one of the sources from the load with the help of the DC/DC converters. The power supplied to the load can be controlled by adjusting the duty cycle of the converter. The configuration is shown in Fig. 2 (b) is preferred due to its robustness, by letting the battery handle the peak load demand. While in the configuration seen in Fig. 2 (a), the system is limited by the peak current allowed by the duty cycle of the DC/DC converter, and also by the amount of energy stored in the supercapacitor.

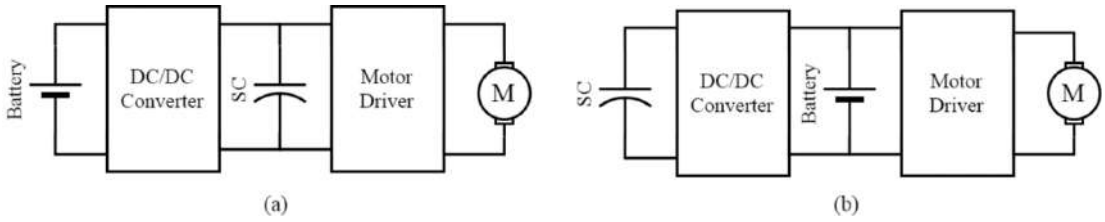

Figure 2: Semi-active configuration

Multiple DC/DC converters could be placed in the above systems to increase system flexibility; such configurations are called full active configurations, as shown in Fig. 3. The full active configuration decouples all sources from the load, and the sources could connect in series as seen in Fig. 3 (a) where the battery is intended to supply the average load, whereas peak power requirements are supplied by the supercapacitor. On the other hand, the parallel combination offers the highest flexibly due to the ability to control each source somewhat independently. Generally, managing the system power with multiple converters is considered a complicated task due to a need for controlling different source simultaneously [4].

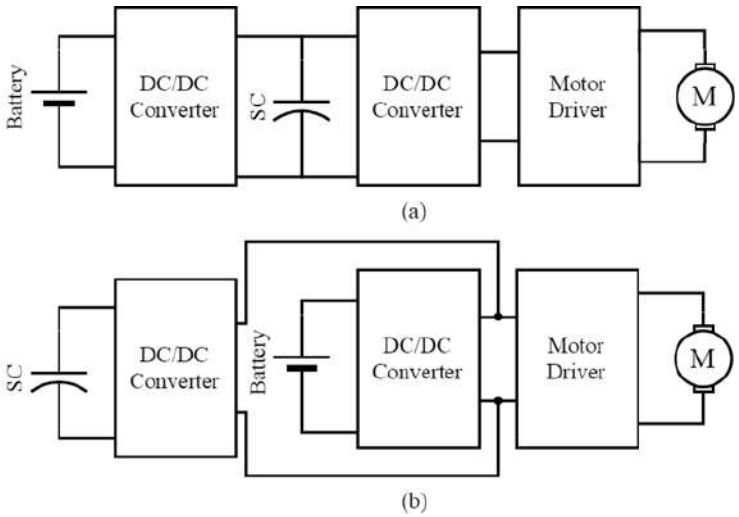

Figure 3: Series/parallel full active configuration

1.2.2. Power management strategies. Power management strategies have been categorized into two major types; rule-based and optimization-based techniques. The techniques are based on some prior knowledge processed through if-then statements or look-up tables. In [8], the power demand is divided between a battery and internal combustion engine based on vehicle acceleration and battery state of charge (SOC), by connecting/disconnecting internal combustion engine/battery to the load, or simultaneously using both of them. Another rule-based technique is applied in [9] by defining a static power threshold for the battery. As an extension of the traditional rule-based approach, fuzzy logic can be used to deal with complex decisions. The fundamental

idea of this technique is about using the known information about the system in order to form a set of fuzzy rules close to the human way of thinking in terms of approximations. The main advantage of the fuzzy technique is the robustness to the variation of different input measurements, as well as, its simplicity in terms of rules formulation and tuning. In [10], a fuzzy logic approach is applied in order to manage the energy of a bus powered by a battery, supercapacitor, and fuel cell. The controller was able to maintain battery state of charge (SOC) within specific boundaries with the help of multiple input DC/DC converter.

The power can be optimally managed through optimizing a specific cost function over a known driving cycle. Different optimization approaches like dynamic programming (DP), linear programming, and convex optimization have been used in this area. In [11], the DP approach is applied in order to find out the optimal path for a vehicle that guarantees minimum fuel consumption. Usually, the optimization is implemented offline because it is computationally expensive. The driving cycle may not be completely known for electric vehicles and mobile robots. An artificial neural network (ANN) can be used to deal with such uncertainty [12]. The work published in [13] considers a power management problem for a hybrid electric vehicle based on model predictive control (MPC). In [14], a model predictive control-based approach was simulated for a battery, fuel cell, supercapacitor system in order to find out the proper reference current to be fed to proportional-integral controllers which generate the duty cycles of a DC/DC converters. In [15], the objectives of the optimization problem in MPC were to minimize the current fluctuation, as well as, reduce the supercapacitor voltage variations from the reference voltage.

1.2.3. Model predictive control. The permanent magnet DC (PMDC) motors have been widely investigated, and several superior control approaches have been introduced in the literature. Where the simplest way to control a PMDC machine is hysteresis control, the main disadvantage of hysteresis control is the huge power loss due applying positive and negative maximum source voltage across PMDC machine terminals. Among the new approaches, an attractive way of controlling permanent magnet DC motor is model predictive control (MPC).

The acronym Model-based Predictive Control (MPC) denotes different types of predictive control laws, e.g. IDCON [16], DMC [17], GPC [18], QDMC [19], IMC [20]. MPC philosophically imitates human reaction, whereby control actions are chosen, which are assumed to reach to the best-predicted output over certain horizon. The internal model of the process (plant) is used to determine the horizon range. Plant model must reflect the relationship between the output and current/future inputs. The model is formulated as a transfer function in case of single input single output system (SISO) or state-space approaches may be used in case of multi-inputs multi-output (MIMO) model. Practically, the majority of MPC algorithms are formulated based on linear models. However, nonlinear models can still be used with MPC, but the computational time and model approximations are the main difficulties. Since predictive control uses the model solely to predict system output, the work done during the modeling stage should show reliable system predictions. The system model is used to get an approximate idea future information from the faraway horizon in order to update MPC controller control actions (decisions).

1.2.4. Battery terminal voltage collapse detection. Battery behavior and certain properties start to deviate from the normal conditions when a battery is about to die. Different methods may be used to predict or detect battery terminal voltage collapse. Using a voltage threshold (VT), or capacity threshold (CT) are simple methods to switch batteries when a battery terminal voltage is about to drop below operational requirements. However, a threshold does not provide pre-warning of impending battery terminal voltage collapse. If an additional VT or CT is used for battery terminal voltage collapse pre-warning system, then several issues need to be considered. For example, a VT [21] can lead to false alarms in existence of momentary surge discharge currents. Similarly, CT [22] can be used to create a threshold on battery state of charge (*SOC*) to make a battery terminal voltage collapse pre-warning system. But the CT approach may be highly influenced by the accuracy of battery current measurement, number of charging/discharging cycles, and temperature. The most commonly used technique to calculate battery state of charge is Coulomb counting, which involves battery current integration [23]. The main concerns of this method are, the initial battery *SOC*, and

inaccurate current measurements. In [24], a rule-based and probabilistic method is applied to detect battery voltage collapse. However, rules mentioned in [24] usually capture only simple changes in battery behavior, and probabilistic approaches require knowledge of a probability density function, which is hard to get. The approach in [25] uses battery impulse response, and look-up tables to detect battery failure. But, look-up tables are battery specific and need substantial effort to develop. Also, if battery characteristics change, then the look-up tables need to be obtained again. According to [26], using a battery model enhances battery fault detection accuracy. Further, battery *SOC* can be estimated with the help of a Kalman filter (KF) or an extended Kalman filter (EKF) as discussed in [27]. Filtering approaches such as KF and EKF require a detailed battery model. However, obtaining an accurate battery model and its parameters can be challenging [28], [29], and [30]. In [31], different tests are performed to show how VT and CT based battery terminal voltage collapse detection methods can be inadequate, and this is fixed with help of adaptive threshold (AT). In [32], battery terminal voltage collapse is detected without a sophisticated model based on universal adaptive stabilizer (UAS), however, this approaches trend detector, and picking window sizes for a trend detector can be complex.

1.3. Motivation

Form the proposed literature review, it is apparent that the power management problem for mobile robots powered by multiple energy sources, still has ample room for innovation. Based on this observation, this work proposes to develop a passive strategy for a hybrid energy system on board a mobile robot, and Generalized Predictive Control (GPC) combined with a Kalman filter (KF). In addition, this work proposes a novel battery terminal voltage collapse detection method based on fast Fourier transform (FFT) of battery terminal voltage, and then processing this data through an artificial neural network (ANN). The developed battery failure prediction/detection method will be used in real-time to find out the right moment to switch one battery out of the job of supplying power to the mobile robot, and letting the backup battery handle the load demand.

1.4. Thesis Organization

Chapter 2 focuses on the mathematical background and gives a brief explanation of the models used. Chapter 3 discusses the robot chassis, hardware setup, and different components specifications. In Chapter 4, preliminary work is discussed. In Chapter 5, the results for this work are introduced. In Chapter 6, the conclusion is presented to summarize the main outcomes achieved.

Chapter 2: Background

2.1. Chen and Mora's Model

Chen and Mora's model [28], applied in this work, is shown in Fig. 4 in order to model the battery unit. The left side of the model, shown in Fig. 4, represents the battery state of charge (*SOC*) behavior, while the right side describes the battery output voltage with respect to the variation of load current. While the state denoted by x_1 represents the battery *SOC* as discussed, the state denoted by x_2 represents the voltage drop across $R_{ts}\|C_{ts}$, and the state denoted by x_3 represents the voltage drop across $R_{tl}\|C_{tl}$. The parallel combination $R_{ts}\|C_{ts}$ models the short-term terminal voltage behavior with respect to the variation of the discharge current. Likewise, the parallel combination $R_{tl}\|C_{tl}$ shows the long-term terminal voltage dynamics with respect to the variation of the discharge current: the state $x_1 \in [0,1]$, and the state x_2,x_3 in R. The state space equations for Chen and Mora's model as derived in [33] are:

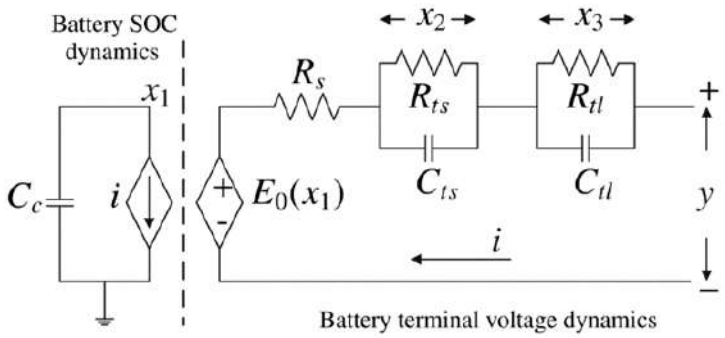

Figure 4: Chen and Mora's battery model

$$\dot{x}_1(t) = -\frac{1}{C_c} i(t), \;\; C_c = 3600 C f_1 f_2 \tag{1}$$

$$\dot{x}_2(t) = -\frac{x_2(t)}{R_{ts}(x_1) C_{ts}(x_1)} + \frac{i(t)}{C_{ts}(x_1)} \tag{2}$$

$$\dot{x}_3(t) = -\frac{x_3(t)}{R_{tl}(x_1) C_{tl}(x_1)} + \frac{i(t)}{C_{tl}(x_1)} \tag{3}$$

$$y(t) = E_o(x_1) - x_2(t) - x_3(t) - i(t) R_s(x_1) \tag{4}$$

Where the components R_{ts}, R_{tl}, C_{ts}, C_{tl}, R_s and E_o are given as follows:

$$R_{ts}(x_1) = k_7 e^{-k_8 x_1} + k_9 \tag{5}$$

$$R_{tl}(x_1) = k_{10} e^{-k_{11} x_1} + k_{12} \tag{6}$$

$$C_{ts}(x_1) = -k_{13} e^{-k_{14} x_1} + k_{15} \tag{7}$$

$$C_{tl}(x_1) = -k_{16} e^{-k_{17} x_1} + k_{18} \tag{8}$$

$$E_o(x_1) = -k_1 e^{-k_2 x_1} + k_3 + k_4 x_1 - k_5 x_1^2 + k_6 x_1^3 \tag{9}$$

$$R_s(x_1) = k_{19} e^{-k_{20} x_1} + k_{21} \tag{10}$$

R_s, R_{ts}, R_{tl}, C_{ts}, and C_{tl} refer to the resistances and capacitances of Chen Mora's model shown in Fig. 4. The term E_o expresses the open circuit voltage of a Li-ion battery. $k_i > 0$ for $i = \{1, 2, 3, \ldots 21\}$ are the battery model parameters that required the identification given in [29] and [30]. Further, the discussed battery model is constructed in MATLAB/Simulink environment as shown in Fig. 5.

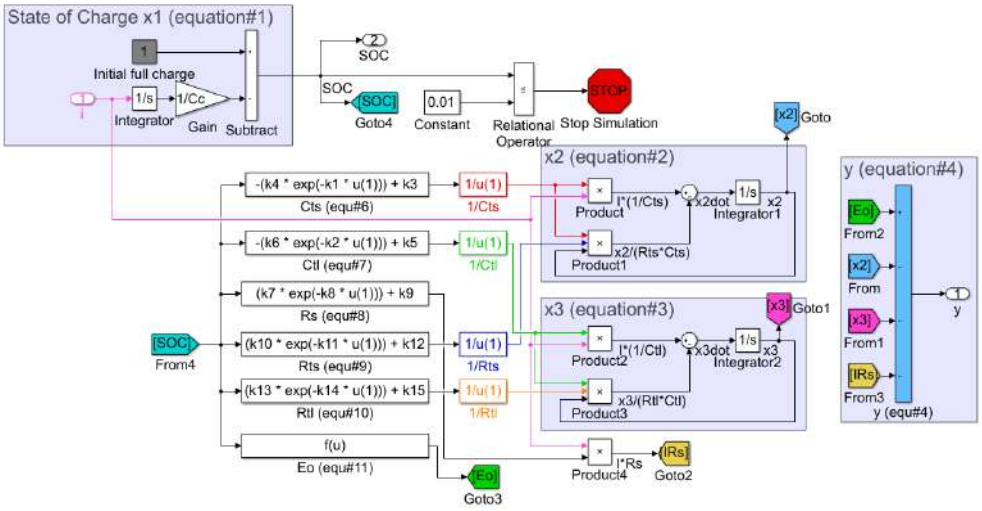

Figure 5: Chen and Mora's model in MATLAB/Simulink environment

2.2. Continuous and Discrete PMDC Motor Model

In this section, a continuous and discrete model for PMDC motor are formulated. The electrical and mechanical dynamics of PMDC motor are represented by Eq.

(11) and Eq. (12), respectively.

$$v_a = R_a i_a + L_a \frac{di_a}{dt} + K_b \omega_m \tag{11}$$

$$\frac{d\omega_m}{dt} = \frac{K_t}{J_{eq}} i_a - \frac{B_m}{J_{eq}} \omega_m - \frac{1}{J_{eq}} T_L \tag{12}$$

Where R_a is the armature resistance, L_a is the armature inductance, i_a and v_a are motor armature current and terminal voltage, respectively. K_t and K_b are the motor torque constant (Nm/A) and the back EMF constant (Nm/rad/s), respectively. Where J_{eq} and B_m are the motor equivalent inertia and viscous friction, respectively. Also ω_m is the shaft rotational speed (rad/sec), and T_L is the torque load (Nm). Note that, an approximation is used by setting T_L to zero since the motor is identified under no-load condition. The above system equations can be represented in state-space form as follows:

$$\frac{d\mathbf{x}(t)}{dt} = \mathbf{A_c}\mathbf{x}(t) + \mathbf{B_c}u(t) + \mathbf{v}(t); \quad \mathbf{y}(t) = \mathbf{C_c}\mathbf{x}(t) + \mathbf{w}(t) \tag{13}$$

$$\textit{where}, \ \mathbf{x} = \begin{bmatrix} \omega_m \\ i_a \end{bmatrix}, \ u = v_a, \ \mathbf{A_c} = \begin{bmatrix} -\frac{B_m}{J_{eq}} & \frac{K_t}{J_{eq}} \\ -\frac{K_b}{L_a} & -\frac{R_a}{L_a} \end{bmatrix}, \ \mathbf{B_c} = \begin{bmatrix} 0 \\ \frac{1}{L_a} \end{bmatrix}, \ \mathbf{C_c} = \begin{bmatrix} 1 & 0 \end{bmatrix}.$$

$\mathbf{A_c}$, $\mathbf{B_c}$, and $\mathbf{C_c}$ are the system dynamics, input, and measurement matrix, respectively. Furthermore, $\mathbf{v}(t)$ and $\mathbf{w}(t)$ represent the process and measurement noise, and they are in form of uncorrelated Gaussian random variables with zero mean. The continuous time model is approximated through Euler approximation to get the discrete time system model given in Eq. (14).

$$\mathbf{x}_{k+1} = \mathbf{A_d}\mathbf{x}_k + \mathbf{B_d}u_k + \mathbf{v}_k; \quad \mathbf{y}_k = \mathbf{C_d}\mathbf{x}_k + \mathbf{w}_k \tag{14}$$

$$\textit{where}, \ \mathbf{A_d} = \begin{bmatrix} 1 - (\frac{B_m}{J_{eq}})\Delta & \frac{K_t}{J_{eq}}\Delta \\ -\frac{K_b}{L_a}\Delta & 1 - (\frac{R_a}{L_a})\Delta \end{bmatrix}, \ \mathbf{B_d} = \begin{bmatrix} 0 \\ \frac{\Delta}{L_a} \end{bmatrix}, \ \mathbf{C_d} = \begin{bmatrix} \mathbf{C_c} \end{bmatrix}.$$

2.3. PMDC Motor Identification

As introduced in section 2.2, the motor model is linear time-invariant (LTI) model, and the PMDC motor parameters needs to be found with sufficient accuracy to

do predictions. As discussed in [34], motor parameters are heat dependent, meaning, an accurate model is formed by nonlinear equations with time varying parameters. However, GPC does not require an accurate model since it updates its predictions in each time step (Δ). Therefore, approximated PMDC machine equations given in Eq. (11) and Eq. (12) are suitable to accomplish such a task. In [35], the electrical parameters (K_b, R_a, La) within Eq. (11) can be determined through the open-circuit, locked rotor tests. Likewise, motor mechanical parameters (B_m, J_{eq}) within Eq. (12) are obtained based on the work discussed in [36].

2.3.1. Evaluating motor back EMF (K_b) and torque constant (K_t). Motor back EMF constant (K_b) can be determined from the direct relationship between the back EMF (e_a) and motor speed as given in Eq. (15). A number of motor speeds in rad/sec and back EMF (e_a) across PMDC machine terminals are measured, and the slope of Eq. (15) is found since it represents the back EMF (K_b). Furthermore, back EMF constant (K_b) and torque constant (K_t) are assumed to be the same value, since they are having approximately the same value in the literature.

$$e_a = K_b \omega_m \tag{15}$$

2.3.2. Determining motor resistance (R_a). The locked rotor test is applied to identify the motor resistance (R_a) and inductance (L_a) experimentally. When the rotor is locked, both motor angular speed ω_m and the rate of change of motor current $\frac{di_a}{dt}$ are set to zero. Therefore, the Eq. (11) results in the simple form given Eq. in (16). Further, both of v_a and i_a are measured, and then R_a can be simply determined.

$$v_a = R_a i_a \tag{16}$$

2.3.3. Finding motor inductance (L_a). Motor inductance can be determined close to the instant that the motor terminal voltage is applied; at this moment, the produced EMF is zero (no angular speed yet), and the motor current is approximately zero. Just at the moment of applying the voltage to the motor coil, the inductance slows down

the change in current values. As a result, the Eq. (11) is reduced to Eq. (17).

$$v_a = L_a \frac{di_a}{dt} \tag{17}$$

2.3.4. Obtaining motor viscous friction (B_m). In this section, the viscous friction (B_m) of a PMDC is obtained under no-load steady-state case in order to drive ($J_{eq}\frac{d\omega_m}{dt}$) and ($T_L$) to zero. As a result, the mechanical Eq. (12) is simplified to Eq. (18). In Table 11, both PMDC motor speed and current are considered under no-load (NL), steady-state (SS) condition, and listed. Since K_t is known, then B_m can be calculated.

$$0 = K_t(i_a)_{SS}^{NL} - B_m(\omega_m)_{SS}^{NL} \tag{18}$$

2.3.5. Calculating motor equivalent inertia (J_{eq}). In this section, the moment of inertia (J_{eq}) is found under no-load steady-state condition to drive ($J_{eq}\frac{d\omega_m}{dt}$) and ($T_L$) to zero. In the first stage, the PMDC motor is mentioned at constant speed, and then the supplied power is switched off from the PMDC motor to let ($K_t i_a$) go to zero. As a result, the mechanical Eq. (12), just after shutting down the motor (t_0^+), is simplified to Eq. (19).

$$J_{eq}(\frac{d\omega_m}{dt})_0^+ = K_t i_a - B_m(\omega_m)_0^+ \tag{19}$$

2.4. Low Level Control - Generalized Predictive Control (GPC)

In this section, a linear MPC strategy, called Generalized Predictive Control (GPC), unconstrained case, is illustrated. Usually, GPC is performed in discrete-time as discussed in [37].

2.4.1. CARIMA model. A general representation [38] and [39], known as controlled auto-regressive integrated moving average (CARIMA), in form of a discrete

transfer function combined with a disturbance term is given in Eq. (20).

$$a(z)y_k = b(z)u_k + \frac{T(z)}{\Delta}v_k \tag{20}$$

Where v_k represents the measurement noise by a Gaussian random variable with zero mean, $T(z)$ is a filtering term in the form of transfer function within the GPC algorithm. Further, the work in [38] and [39] illustrate the way of designing $T(z)$ term because it has an impact on system close-loop performance. While $a(z)$ and $b(z)$ are the denominator and numerator coefficients of the system transfer function, respectively. Further, system output and input are represented by y_k and u_k, respectively. Finally, system time step is denoted by Δ.

Usually, the transfer function can be written in form of an equivalent difference equation to obtain Eq. (21), where $a(z)$, $b(z)$, and d_k are represented by Eq. (22), Eq. (23), and Eq. (24), respectively. The obtained transfer function will be used in the next subsection to do predictions.

$$y_k + a_1 y_{k-1} + ... + a_n y_{k-n+1} = b_1 u_{k-1} + b_2 u_{k-2} + ... + b_n u_{k-n+1} + d_k \tag{21}$$

$$a(z) = 1 + a_1 z^{-1} + a_2 z^{-2} + ... + a_n z^{-n} \tag{22}$$

$$b(z) = b_1 z^{-1} + b_2 z^{-2} + ... + b_n z^{-n} \tag{23}$$

$$d_k = \frac{T(z)}{\Delta}v_k \tag{24}$$

2.4.2. Prediction with CARIMA model. The approach discussed in [37] is related to the case of $T(z)$=1. The incremental CARIMA model is given by Eq. (25). Then, let $A(z) = a(z)\Delta$ is defined, and the incremental CARIMA model can be reformed as given in Eq. (26), where $A(z)$ and $b(z)$ are given by Eq. (27), Eq. (23), respectively.

$$a(z)\Delta y_k = b(z)\Delta u_k \tag{25}$$

$$A(z)y_k = b(z)\Delta u_k \tag{26}$$

$$A(z) = 1 + A_1 z^{-1} + A_2 z^{-2} + ... + A_{n+1} z^{-n-1} \tag{27}$$

$$y_{k+1} = -A_1 y_k - ... - A_n y_{k-n+1} + b_1 u_k + ... + b_n u_{k-n+1} \tag{28}$$

$$y_{k+1} = -[A_1, ..., A_n] \underleftarrow{y_k} + [b_2, ..., b_n] \underleftarrow{\Delta u_{k-1}} + b_1 \Delta u_k \tag{29}$$

$$y_{k+2} = -A_1 y_{k+1} - ... - A_n y_{k-n+1} + b_1 u_{k+1} + b_2 y_k + ... + b_n u_{k-n+2} \tag{30}$$

$$y_{k+n_y} = -A_{n+1} y_{k+n_y+1-n} - ... - A_n y_{k-n+1} + b_1 u_{k+n_y-1} + ... + b_n u_{k+n_y-n} \tag{31}$$

The difference equation given by Eq. (21) can be manipulated to form Eq. (28), where the idea behind this manipulation is to have one-step ahead prediction because of the term y_{k+1}. Then Eq. (28) can be re-arranged further to predict future output y_{k+1} based on current and past input increment (Δu_k), in addition to the current output (y_k) as given in Eq. (29). Same procedures can be used to predict two steps ahead through substituting Eq. (28) into Eq. (30). As a result, Eq. (31) can be used to predict n-steps a head, where n_y denotes the horizon prediction range. The form shown in Eq. (27) can be lumped to Eq. (33). The predicted output $\underrightarrow{y_k}$ is given by Eq. (34), furthermore, $\underrightarrow{y_k}$ can be re-arranged as shown in Eq. (35); where $H = {C_A}^{-1} C_b$, $P = {C_A}^{-1} H_b$, and $Q = {C_A}^{-1} H_A$.

$$\underbrace{\begin{bmatrix} 1 & 0 & \dots & 0 \\ A_1 & 1 & \dots & 0 \\ A_2 & A_1 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}}_{C_A} \underbrace{\begin{bmatrix} y_{k+1} \\ y_{k+2} \\ \vdots \\ y_{k+n_y} \end{bmatrix}}_{\underrightarrow{y_k}} + \underbrace{\begin{bmatrix} A_1 & A_2 & \dots & A_{n+1} \\ A_2 & A_3 & \dots & 0 \\ A_3 & A_4 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}}_{H_A} \underbrace{\begin{bmatrix} y_k \\ y_{k-1} \\ \vdots \\ y_{k-n} \end{bmatrix}}_{\underleftarrow{y_k}}$$

$$= \underbrace{\begin{bmatrix} b_1 & 0 & \dots & 0 \\ b_2 & b_1 & \dots & 0 \\ b_3 & b_2 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}}_{C_b} \underbrace{\begin{bmatrix} \Delta u_k \\ \Delta u_{k+1} \\ \vdots \\ \Delta u_{k+n_y-1} \end{bmatrix}}_{\underrightarrow{\Delta u_k}} + \underbrace{\begin{bmatrix} b_2 & b_3 & \dots & b_n \\ b_3 & b_4 & \dots & 0 \\ b_4 & b_5 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix}}_{H_b} \underbrace{\begin{bmatrix} \Delta u_{k-1} \\ \Delta u_{k-2} \\ \vdots \\ \Delta u_{k-n+1} \end{bmatrix}}_{\underleftarrow{\Delta u_{k-1}}} \tag{32}$$

$$C_A \underset{\rightarrow}{y_k} + H_A \underset{\leftarrow}{y_k} = C_b \underset{\rightarrow}{\Delta u_k} + H_b \underset{\leftarrow}{\Delta u_{k-1}} \tag{33}$$

$$\underset{\rightarrow}{y_k} = {C_A}^{-1}[C_b \underset{\rightarrow}{\Delta u_k} + H_b \underset{\leftarrow}{\Delta u_{k-1}} - H_A \underset{\leftarrow}{y_k}] \tag{34}$$

$$\underset{\rightarrow}{y_k} = H \underset{\rightarrow}{\Delta u_k} + P \underset{\leftarrow}{\Delta u_{k-1}} + Q \underset{\leftarrow}{y_k} \tag{35}$$

2.4.3. Cost function and optimization. The control law is defined by minimizing the cost function (**J**) *w.r.t*. the future control increment $\underset{\rightarrow}{\Delta u_k}$. As the cost function is quadratic, it has a single minimum that can be reached by applying the gradient. As given in Eq. (36), the cost function (**J**) should cover system performance till time tends to infinity; however, it can be approximated using a prediction horizon (n_y) with value greater than plant settling-time as discussed in [37]. Further, Eq. (36) can be expanded to form Eq. (37) since the error is represented by $\underset{\rightarrow}{r}_k - \underset{\rightarrow}{y}_k$, where $\underset{\rightarrow}{r}_k$ and $\underset{\rightarrow}{y}_k$ are denoted the reference and plant measured output, respectively. The given predicted output $\underset{\rightarrow}{y_k}$ by Eq. (35) is substituted into Eq. (37), as a result of this substitution, Eq. (38) is formulated after finalizing the substitution, where this equation needs to be minimized *w.r.t*. $\underset{\rightarrow}{\Delta u_k}$. According to algebra, the gradient of $(x^T a)$ is equal to a. Furthermore, applying the gradient to $(x^T S x)$ results in $(S + S^T)x$. The previous shortcuts are used to minimize the cost function in Eq. (38), where the result of the minimization is shown in Eq. (40).

$$\mathbf{J} = \sum_{k=0}^{\infty} (\underset{\rightarrow}{e}_k)^2 + \lambda (\underset{\rightarrow}{\Delta u_k})^2 \approx \sum_{k=0}^{n_y} (\underset{\rightarrow}{e}_k)^2 + \lambda (\underset{\rightarrow}{\Delta u_k})^2 \tag{36}$$

$$\mathbf{J} = \sum_{k=0}^{n_y} (\underset{\rightarrow}{r}_k - \underset{\rightarrow}{y}_k)^T (\underset{\rightarrow}{r}_k - \underset{\rightarrow}{y}_k) + \lambda (\underset{\rightarrow}{\Delta u_k})^T (\underset{\rightarrow}{\Delta u_k}) \tag{37}$$

$$0 = \min_{\underset{\rightarrow}{\Delta u_k}} \ \mathbf{J} \tag{38}$$

$$0 = grad_{\underset{\rightarrow}{\Delta u_k}} \underbrace{\left(\underbrace{\underset{\rightarrow}{\Delta u_k}^T}_{x^T} \underbrace{(H^T H + \lambda I)}_{S} \underbrace{\underset{\rightarrow}{\Delta u_k}}_{x} - \underbrace{\underset{\rightarrow}{\Delta u_k}^T}_{x^T} \underbrace{(2H^T)(\underset{\rightarrow}{r}_k - P \underset{\leftarrow}{\Delta u_{k-1}} - Q \underset{\leftarrow}{y}_k)}_{a} \right)}_{\mathbf{J}} \tag{39}$$

$$0 = 2(H^T H + \lambda I)\underset{\rightarrow}{\Delta u_k} - (2H^T)(\underset{\rightarrow}{r}_k - P \underset{\leftarrow}{\Delta u_{k-1}} - Q \underset{\leftarrow}{y}_k) \tag{40}$$

In this stage, the Eq. (41) is re-arranged *w.r.t.* $\underrightarrow{\Delta u}_k$ to give a set of predicted control increments $[\Delta u_{k-1}\ \Delta u_{k-2}\ \ldots\ \Delta u_{k-n+1}]^T$. Then, Eq. (41) is multiplied by vector $E = [I\ 0\ \ldots\ 0]^T$, where this vector is used to get the first element from each predicted set as given in Eq. (42).

$$\underrightarrow{\Delta u}_k = (H^T H + \lambda I)^{-1} H^T (\underrightarrow{r}_k - P \underleftarrow{\Delta u}_{k-1} - Q \underleftarrow{y}_k) \tag{41}$$

$$\underrightarrow{\Delta u}_k = \underbrace{E^T (H^T H + \lambda I)^{-1} H^T}_{P_r} (\underrightarrow{r}_k - P \underleftarrow{\Delta u}_{k-1} - Q \underleftarrow{y}_k) \tag{42}$$

Finally, P_r term is used to reduced Eq. (42) to Eq. (43), where P_r is nothing more than $E^T (H^T H + \lambda I)^{-1} H^T$. In Eq. (43), new matrices D and N are used for further simplification, where D and N are represented by $[P_r P]$ and $[P_r Q]$, respectively. As a result, the final control law is found in Eq. (44) and its diagram is shown by Fig.6.

$$\underrightarrow{\Delta u}_k = P_r \underrightarrow{r}_k - \underbrace{P_r P}_{D} \underleftarrow{\Delta u}_{k-1} - \underbrace{P_r Q}_{N} \underleftarrow{y}_k \tag{43}$$

$$\underrightarrow{\Delta u}_k = P_r \underrightarrow{r}_k - D \underleftarrow{\Delta u}_{k-1} - N \underleftarrow{y}_k \tag{44}$$

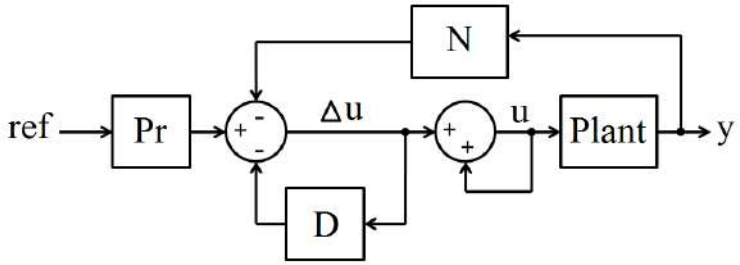

Figure 6: GPC diagram

2.5. State Observer

In this section, the formulation of a discrete Kalman filter (KF) is used to estimate PMDC motor angular speed is discussed. As reported in [37], filtering approaches are a powerful tool to enhance the overall performance of GPC controllers. The discrete time form of a PMDC motor model is represented by Eq. (14), where the system states

are $x = [\omega_m \; i_a]^T$. The process noise covariance matrix is given by Eq. (45). Likewise, measurement noise covariance matrix is given in Eq. (46).

$$\mathbf{Q} = diag[q_{11},\ q_{22}] \tag{45}$$

$$\mathbf{R} = [r_{11}] \tag{46}$$

According to [40], the KF algorithm consists of a prediction followed by an update step, where the predication (priori estimate) is represented by Eq. (47) and Eq. (48). Note that, the dynamic ($\mathbf{A_d}$) and input matrix ($\mathbf{B_d}$) are defined in section (2.2).

$$\hat{\mathbf{x}}_{k+1}^{-} = \hat{\mathbf{x}}_{k}^{+}\mathbf{A_d} + \mathbf{B_d}\mathbf{u} \tag{47}$$

$$\mathbf{P}_{k+1}^{-} = \mathbf{A_d}\mathbf{P}_k\mathbf{A_d^T} + \mathbf{Q} \tag{48}$$

The update process starts by computing Kalman filter gain ($\mathbf{K}$) given in Eq. (49). According to [41], $\mathbf{K}$ will converge to a steady-state value when the used model is linear. Further, the corrected state estimate ($\hat{\mathbf{x}}_{k+1}^{+}$) is obtained based on the priori estimate ($\hat{\mathbf{x}}_{k+1}^{-}$), Kalman filter gain ($\mathbf{K}$), and the sampled measurement (z_{k+1}) as shown in Eq. (50). Finally, the updated covariance ($\mathbf{P}_{k+1}^{+}$) is calculated based on priori covariance ($\mathbf{P}_{k+1}^{-}$) and $\mathbf{K}$ as given in Eq. (51).

$$\mathbf{K}_{k+1} = \mathbf{P}_{k+1}^{-}\mathbf{C_d^T}(\mathbf{C_d}\mathbf{P}_{k+1}^{-}\mathbf{C_d^T} + \mathbf{R})^{-1} \tag{49}$$

$$\hat{\mathbf{x}}_{k+1}^{+} = \hat{\mathbf{x}}_{k}^{-} + \mathbf{K}_{k+1}(z_{k+1} - \mathbf{C_d}(\hat{\mathbf{x}}_{k+1}^{-})) \tag{50}$$

$$\mathbf{P}_{k+1}^{+} = (I - \mathbf{K}_{k+1}\mathbf{C_d})\mathbf{P}_{k+1}^{-} \tag{51}$$

2.6. Differential-drive Mobile Robot Model

2.6.1. Odometry model. The model is formulated for the differential-drive robot seen in Fig. 7, where this model uses measured parameters (e.g. robot dimensions) to relate model inputs to the outputs without including detailed parameters.

Robot dimensions are represented by wheel radius (r) and the distance between the wheels (L). The inputs to the model are represented by the right wheel speed denoted by (v_R), and left wheel speed denoted by (v_L). Typically, the model shows robot position (x, y) and heading (θ); as a result, the model tells us where the robot is on a plant using (x, y) and direction it is going based on θ. Furthermore, Eq. (52), Eq. (53), and Eq. (54) are formulated so as to show the relation between model states $[x \;\; y \;\; \theta]^T$ and its inputs (v_R, v_L). However, the model is offering no knowledge about robot linear (v) and rotational speed (ω), which are commonly used to control robot motion. Therefore, unicycle model will be introduced to show the relation between robot states and (v, ω) as inputs.

$$\dot{x} = \frac{r}{2}(v_R + v_L)\cos(\theta) \tag{52}$$

$$\dot{y} = \frac{r}{2}(v_R + v_L)\sin(\theta) \tag{53}$$

$$\dot{\theta} = \frac{r}{L}(v_R - v_L) \tag{54}$$

2.6.2. Unicycle model. Different from Odometry model, the inputs in this model are represented by linear (v) and rotational speed (ω) as seen in Eq. (55), Eq. (56), and Eq. (57). Further, right and left wheel speed are found by equating Odometry to unicycle model equations to each other, and the result of equating are shown in Eq. (58) and Eq. (59).

$$\dot{x} = v\cos(\theta) \tag{55}$$

$$\dot{y} = v\sin(\theta) \tag{56}$$

$$\dot{\theta} = \omega \tag{57}$$

$$v_R = \frac{2v + \omega L}{2r} \tag{58}$$

$$v_L = \frac{2v - \omega L}{2r} \tag{59}$$

2.7. High Level Control - Input/Output State Feedback Linearization

In this section, input/output state feedback linearization (IOSFL) as *high level control* is introduced, where the controller is responsible for generating angular (ω) and linear (v) velocity. According to [42], a point B is placed outside the robot frame with distance b to pull the robot toward the desired point (x_{des}, y_{des}) as shown in Fig. 7. The mathematical representation of IOSFL is initiated by Eq. (60) and Eq. (61). where x and y are the actual robot position, however, the x_b and y_b represent the B point with distance b.

$$x_b = x + b\cos(\theta) \tag{60}$$

$$y_b = y + b\sin(\theta) \tag{61}$$

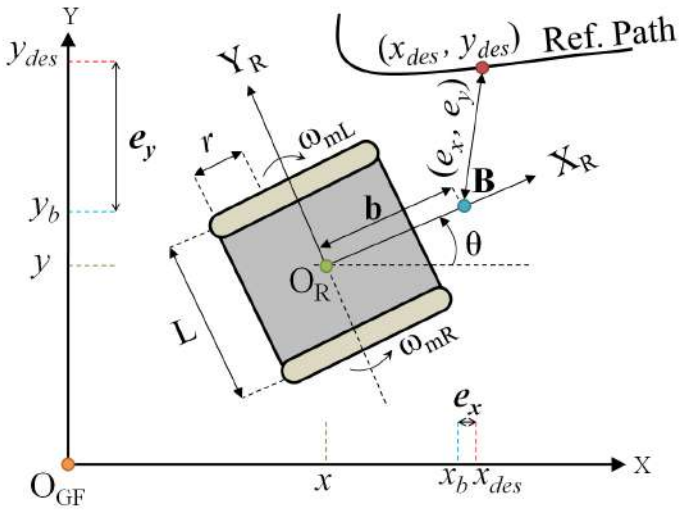

Figure 7: Control problem description

In order to find out v and ω for the robot, the derivative of Eq. (60) and Eq. (61) are found in Eq. (62), Eq. (63), and Eq. (64).

$$\dot{x_b} = v\cos(\theta) - \omega b\sin(\theta) \tag{62}$$

$$\dot{y_b} = v\sin(\theta) - \omega b\cos(\theta) \tag{63}$$

$$\dot{\theta} = \omega \tag{64}$$

Matrix representation is seen in Eq. (65), where v and ω are the inputs to the mentioned equations, while the rate of change $\dot{x}_b$ and $\dot{y}_b$ are the outputs. The inverse of the matrix in Eq. (65) is found since the interest is in v and ω as outputs, where the matrix in Eq. (65) is invertible in case of $b \neq 0$ as shown in Eq. (66).

$$\begin{bmatrix} \dot{x_b} \\ \dot{y_b} \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -b\sin(\theta) \\ \sin(\theta) & b\cos(\theta) \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \tag{65}$$

$$\begin{vmatrix} \cos(\theta) & -b\sin(\theta) \\ \sin(\theta) & b\cos(\theta) \end{vmatrix} = b \neq 0 \tag{66}$$

Therefore, the linear and angular velocity are found in Eq. (67) by inverting the matrix within Eq. (65). Given a predefined path (x_{des}, y_{des}) seen in Fig. 7, it is feasible to obtain $\dot{x}_b$ and $\dot{y}_b$ that ensure asymptotic tracking. This can be implemented using Eq. (68) and Eq. (69), where k_x and k_y are proportional controller gains.

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} \cos(\theta) & \sin(\theta) \\ \frac{-1}{b}\sin(\theta) & \frac{1}{b}\cos(\theta) \end{bmatrix} \begin{bmatrix} \dot{x_b} \\ \dot{y_b} \end{bmatrix} \tag{67}$$

$$\dot{x_b} = k_x e_x \equiv k_x(x_{des} - x_b) \tag{68}$$

$$\dot{y_b} = k_y e_y \equiv k_y(y_{des} - y_b) \tag{69}$$

2.8. Coordinate Transformations

A rotation matrix method is used for reconstructing a vector of measurement from one coordinate representation to another one. The GPS sensor gives the measurement in terms of longitude, latitude, and altitude (LLA), where the robot requires to localize itself based on XYZ–coordinates. Therefore, LLA to Earth–Centered Earth–Fixed (ECEF) Conversion is used to accomplish such a task. Note that, Both od LLA and XYZ–coordinates are found with respect to the origin of the earth.

2.8.1. LLA to XYZ–coordinates conversion in ECEF frame. LLA to XYZ–coordinates in ECEF is implemented based on the equations seen in Eq.(70), Eq.(71), and Eq.(72), where the constants within each equation are discussed in [43]. Further,

Fig. 8 presents the relationship between ECEF and a reference ellipsoid. Where ϕ is the latitude, ψ represents the longitude, h is the height above ellipsoid, and N denotes the radius of curvature.

$$X = (N+h)\cos(\phi)\cos(\psi) \tag{70}$$

$$Y = (N+h)\cos(\phi)\sin(\psi) \tag{71}$$

$$Z = (\frac{b^2}{a^2}N+h)\sin(\phi) \tag{72}$$

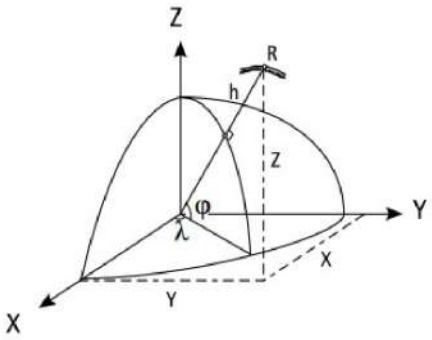

Figure 8: ECEF and reference ellipsoid

2.8.2. XYZ–coordinates in ECEF to ENU conversion. The Conversion from XYZ–coordinates in ECEF to East, North, Up (ENU) coordinates is performed as follows:

$$X_{enu} = \begin{bmatrix} e \\ n \\ u \end{bmatrix} = \begin{bmatrix} -\sin(\psi) & \cos(\psi) & 0 \\ -\cos(\psi)\sin(\phi) & -\sin(\psi)\sin(\phi) & \cos(\phi) \\ \cos(\psi)\cos(\phi) & \sin(\psi)\cos(\phi) & \sin(\phi) \end{bmatrix} \begin{bmatrix} x-x_0 \\ y-y_0 \\ z-z_0 \end{bmatrix} \tag{73}$$

ECEF gives unique LLA or XYZ–coordinates for each position on earth. However, ENU converts the XYZ–coordinates *w.r.t* earth origin to xyz–coordinates *w.r.t* initial robot position. In this work, LLA to XYZ–coordinates is applied firstly, then XYZ–coordinates to ENU is used to compare the GPS XY–measurement to Odometry XY–states.

Chapter 3: Hardware Setup

This chapter describes the hardware setup developed for implementing the hybrid power system based mobile robot.

3.1. Robot Chassis

Virtual robot chassis is shown in Fig. 9. However, the actual robot chassis used in this work is seen in Fig. 10. Furthermore, it is a differential-drive robot with two DC motors. Incremental encoders with resolution (1024 pulses / rotation) are used to measure the rotational speeds of the wheels, which can be used to know the linear and rotational speed of the robot itself. Moreover, a GPS sensor (MIDG II INS/GPS) is used to localize the robot during different tests to drive the robot along a predefined path. In addition, laptop is used to receive all measurements and send control commands to different robot units. Power sources are mounted on the top of the robot chassis as shown in Fig. 9 and Fig. 10.

As a result, battery, fuel cell, and supercapacitor are the sources that are chosen. In addition to this, the aim in our case is to maximize the robot range by applying motion controller capable of finding the minimum amount of input voltage across each motor. Note that, minimum amount voltage drives robot power consumption trowed minimum value.

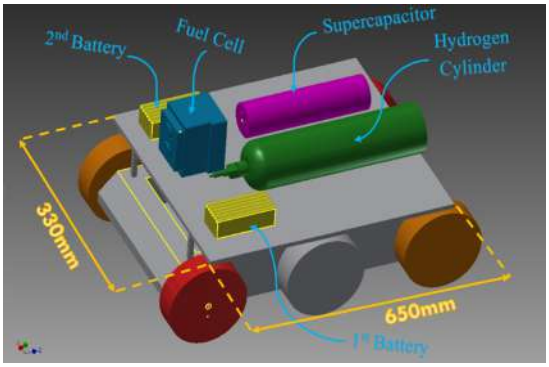

Figure 9: Virtual robot platform.

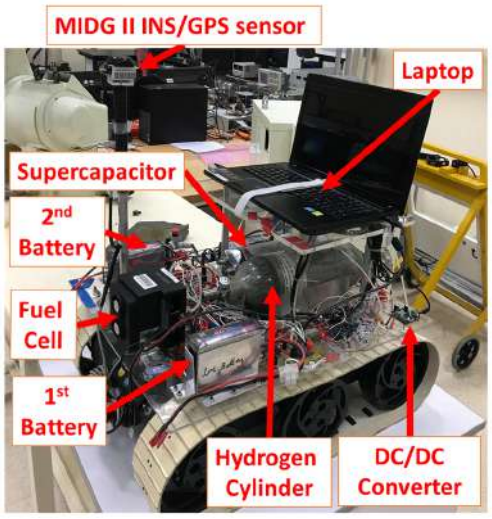

Figure 10: Actual robot platform.

Table 1: Robot Specifications

Robot chassis size	650 mm (L) x 330 mm (W) 300 mm (H)
Wheels radius (R)	0.09 m
distance between wheels (L)	0.59 m
Robot type	Differential
Maximum speed	20 rad/sec
Encoder	YUMO E6B2-CWZ3E incremental encoder with resolution (1024P/R)
PMDC motor	Rated voltage 24V
Chassis weight	10Kg
Max. weight, robot can handle	18Kg

3.2. MIDG II INS/GPS Sensor

A GPS-aided inertial navigation system (INS) is used in localization and navigation applications. An inner GPS receiver (RX) localizes robot position and it then gives this measurement to a fusion processor within the sensor to combine it with the inertial data to generate an optimal solution. MIDG is shown in Fig. 11.

Figure 11: GPS sensor (MIDG II INS/GPS)

3.3. Laptop Computer

Figure 12: Laptop computer

In this thesis, a Lenovo G580 laptop, in Fig. 12, is used as the main processor which receives all measurements and does the computations to control the robot based on different algorithms. The laptop specification is shown in Table 2.

Table 2: Desktop Computer Specification

System Manufacturer	Lenovo
System Model	G580
Processor	Intel Core i5-3630QM @2.40GHz (8CPUs)
RAM	16GB
Graphic Card	AMD Radeon HD 7610M
USB	2 USB 2.0 and 1 USB 3.0

3.4. Motor Driver

In this work, a full H-bridge will be used in order to control the motor efficiently in two directions clockwise and anticlockwise, while the specifications of the motor driver are given in Table 3.

Table 3: Li-ion Battery Specifications

Mpdel	Supply voltage	Continuous of Current	Surge Current	PWM frequency	Current sense output
MegaMoto	5-28V	13A	30A	DC-20kHz	0.0745 V/A

3.5. Li–ion Battery

In this work, two Lithium-ion batteries will be mounted on the robot chassis, where the first battery will be used as main power source and the second battery will be used as a backup source. An ANN-based decision function is proposed to find the right time to switch from the primary battery to the secondary battery.

Table 4: Li-ion Battery Specifications

Model number	Battery capacity	Number of cells	Cell voltage	Battery voltage (nominal)	Battery weight
TP6600-6SP+25	6600 mAH	6	3.7V	22.4V	1Kg

3.6. Fuel Cell

Fuel cells are considered a clean energy source as they are capable of converting one form of energy (hydrogen i.e. chemical) to another form (electricity) with zero emissions. Further, the cells are capable of supplying a load as long as the fuel flow is maintained. Hydrogen is considered as the ideal fuel for fuel cells due to its high energy density compared to other fuel types. In addition to this, the chemical reaction product out of a fuel cell is only water in case of using hydrogen and oxygen with the fuel cell. Comparing to batteries and supercapacitors, fuel cells require specific tank storage on board, and their time response is relatively longer. The specifications of the fuel cell are given in Table 5.

Table 5: PEM fuel cell Specifications

Model	Number of cells	Output voltage	Rated performance	Min. H_2 input pressure	Low voltage shutdown	Weight
Aerostak PEM fuel cell 200W	35	21-32V	10A at 21V	0.55 bar	20V	500g

3.7. Supercapacitor

Comparing to batteries and fuel cells, a supercapacitor has superior performance during transient periods even with heavy loads. The specifications of the supercapacitor proposed to be used, are given in Table 6.

Table 6: Supercapacitor specifications

Rated voltage	Capacity
24V	3F

3.8. Current Sensor

In this work, the current sensor (LEM LA-25 NP) is used to measure different source and load currents, while the specifications of the current sensor are given in Table 7.

Table 7: Current sensor specifications

Model number	Max. input current	Supply voltage	Type
LEM LA-25 NP	25A	± 15	Hall effect

Chapter 4: Experimental Work and Results

4.1. Battery Terminal Voltage Collapse Detection using Fast Fourier Transforms and Neural Networks

As a battery is discharged, its *SOC* will keep decreasing until battery terminal voltage collapse occurs when the low *SOC*. As reported in [32], the aim of any battery terminal voltage collapse method is to detect the transition in battery behavior from stable to the unstable region. However, voltage-dependent methods are suffering from a gradual drop in the measured battery terminal voltage value, and this fact adds more difficulties to detect the true transition when the battery is close to the unstable region (terminal voltage collapse occurs soon). Battery terminal voltage collapse detection methods should preferably overcome any false alarms resulting from gradual or sudden drops in terminal voltage. Artificial neural networks (ANNs) can be trained to accomplish such a task effectively based on the mathematical model of a battery.

The proposed ANN method is compared with conventional battery terminal voltage collapse methods such voltage threshold (VT) and capacity threshold (CT) in terms of the terminal voltage collapse alarm accuracy. Note that, the alarm is considered false alarm (FA) if the used method declared the voltage collapse with *SOC* in the rage of 100 – 10%. The alarm is considered true alarm (TA) if the used method declared the voltage collapse with *SOC* in the rage of 10 – 7%. Finally, the alarm is considered late alarm (LA) if the used method declared the voltage collapse with *SOC* less than 7%.

4.1.1. Training data and feature extraction. This section explains the main procedures applied to create the training data and feature extraction. The process begins with constructing Chen and Mora's battery model [28] in the MATLAB/Simulink environment. Further, model parameters (k_1–k_{21}) are found in [29] and [30] for the same Li-ion battery used in this work. The model is discharged with several loads i to create different battery terminal voltage y profiles. The load currents i were constant, square, sine, triangle waves; and each wave type has different magnitudes (i=1, 2, 3, ... 10A) and different oscillation frequencies (f_{osc}=0.01, 0.1, 1, 10, 100, 1000 Hz). After finalizing the discussed process, a window N is used, where this window consists

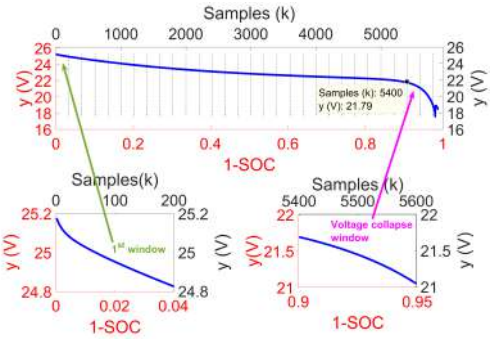

Figure 13: y vs. SOC, 1^{st} window, and a window within collapse region.

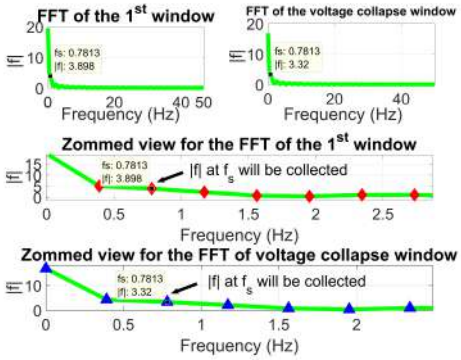

Figure 14: Results of applying FFT for 1^{st}, and voltage collapse window.

of stacked terminal voltage y measurements and N represents the number of measurements within the window. Theoretically speaking, FFT gives a frequency magnitude $|f|$ vs. set of frequencies. As a result of dividing battery terminal voltage to windows with size N, fast Fourier transform (FFT), discussed in [44] is applied for each window to find the frequency magnitude $|f|$ at selected frequency f_s. Note that, the f_s is selected empirically. In this work, $|f|$ is found for f_s = 0.78125Hz. After applying FTT for all y windows, the obtained frequency magnitudes $|f|$ for each window are stacked again in an array to be used as training data for pattern classification, where the target vectors are seen in Fig. 15. One way to accomplish pattern classification is MATLAB toolbox (*nprtool*), where it solves the problem with the help of feed-forward neural network based on sigmoid output neurons according to [45].

To test the above strategy, Chen and Mora's battery model with capacity 6.6Ah is discharged in MATLAB/Simulink environment by constant current i=10A and the sampling time (T_s) is selected as 0.01 Sec. Battery terminal voltage y vs. its capacity SOC is shown in Fig. 13. The window size N is set as 200 samples. The results of the FFT are shown in Fig. 14 for terminal battery voltage y. Furthermore, the frequency magnitude $|f|$ is picked at 0.78125Hz. Generally, the FFT process collects frequency magnitudes at 0.78125Hz for each window until the total number of N is reached. In Fig. 15, FFT is plotted versus battery SOC after finalizing the sliding FFT window process discussed previously for all y. Two target vectors are used to identify the voltage collapse transition moment from stable to unstable region. As seen in Fig. 15, first target vector represents each FFT point in stable region by 0 and all FFT points within unstable

region are labeled as 1s. However, the second target vector is the complement of the first vector, where the FFT points in stable region are labeled as 1s, and the FFT points in unstable region are set to 0s. Based on this, a feed-forward neural network is trained to distinguish between these groups using the FFT data obtained as input feature to ANNs. Chen and Mora's battery model is discharged with different waveforms, several frequencies, and different current magnitudes to create sufficient training data. The proposed classification approach will be discussed further in the next section.

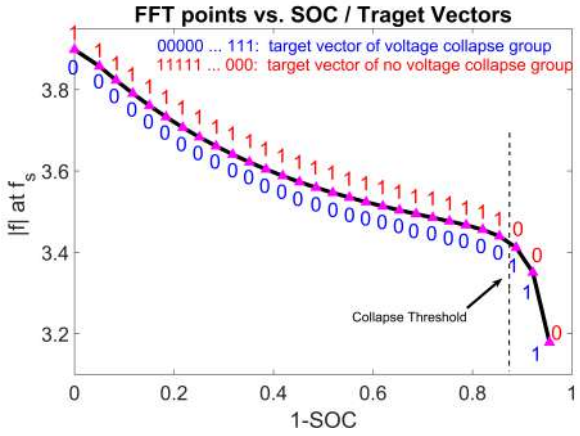

Figure 15: FFT vs. *SOC* in addition to target vector of voltage and no voltage collapse.

4.1.2. Classification based on neural network. As discussed in [46] and [47], an ANN is capable of calculating logical and arithmetic equations. The artificial neuron is shown in Fig. 16, where the inputs are represented by $p_1, p_2, \ldots p_R$; and weighted $w_{1,1}, w_{1,2}, \ldots w_{1,R}$. The total input n can be determined by Eq. (74). Further, Eq. (74) can be lumped to Eq. (75) in a matrix representation, where b is the bias of the shown artificial neuron in Fig. 16. The output of the neuron can by formulated as given in Eq. (76), where f represents linear/non-linear transfer function of n.

$$n = w_{1,1}p_1 + w_{1,2}p_2 + \ldots + w_{1,R}p_R + b \tag{74}$$

$$n = W_p + b \tag{75}$$

$$a = f(W_p + b) \tag{76}$$

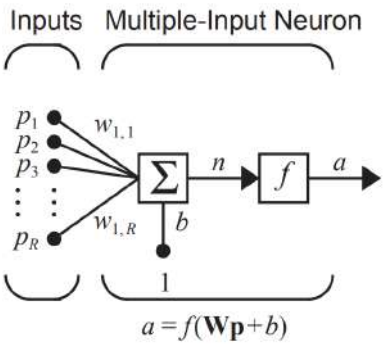

Figure 16: Artificial Neuron [47]

The structure of the neural network is defined by selecting the transfer function f, the number of the inputs, and the neurons. Theoretically, the training is a process to end with acceptable output by updating weights and bias. In this work, the Neural Network Pattern Recognition Toolbox (*nprtool*) is used in the training phase. It supports feed-forward network architecture [45], where the training process of ANN is as follows: First, uploading training data, which it is correctly grouped to the *nprtool* toolbox, second, selecting a number of neurons. and third, specifying one of the training algorithm found in [48].

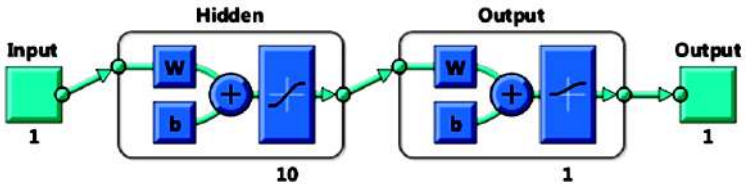

Figure 17: Feed-forward neural network architecture

4.1.3. Feed-forward neural network. The feed-forward neural network is one of the simplest forms categorized under artificial neural networks. As seen in Fig. 17, our architecture consists of a single input node, one hidden layer with n neurons, and two outputs. The input data is processed only in forward direction. It starts from the input node; then it goes through the hidden layer, and it finally ends to the output layer. Note that, there is no feedback loop in this type of neural network. However, a feed-

forward neural network can learn any relation effectively with sufficient training data in addition to enough number of neurons n within the hidden layer.

Table 8: Mean Squared Error (MSE) and Training Time (sec) of Different Training Algorithms

Training algorithm	n=10	n=20	n=30	n=40	n=50	n=60
$trainbfg$	0.069981 17s	0.069209 28s	0.070446 70s	0.069499 75s	0.070621 87s	0.070009 189s
$trainbr$	0.046049 531s	0.04613 776s	0.04629 1535s	0.046109 2072s	0.046392 3002s	0.046168 4107s
$traincgb$	0.069288 24s	0.071194 48s	0.069702 64s	0.06994 91s	0.069743 93s	0.070752 132s
$traincgf$	0.071716 15s	0.070145 30s	0.071217 35s	0.070982 41s	0.069181 57s	0.069482 101s
$traincgp$	0.072076 16s	0.069809 35s	0.071867 61s	0.070933 63s	0.069769 120s	0.0709 205s
$traingd$	0.091464 195s	0.089759 270s	0.092537 212s	0.080721 380s	0.087284 441s	0.083712 501s
$traingdm$	0.093095 203s	0.093252 259 s	0.082842 334s	0.090658 435s	0.076888 469s	0.086937 528s
$traingda$	0.07233 31s	0.071782 44s	0.072897 52s	0.074745 49s	0.074359 61s	0.07671 74s
$traingdx$	0.072954 36s	0.070361 54s	0.070073 60s	0.091935 11s	0.07118 78s	0.070968 90s
$trainlm$	0.046936 74s	0.044938 13s	0.047792 14s	0.04525 39s	0.044737 33s	0.044847 86s
$trainoss$	0.071905 43s	0.069747 80s	0.070736 94s	0.071073 141s	0.07114 134s	0.071696 111s
$trainrp$	0.071935 22s	0.069934 8s	0.06993 17s	0.070492 15s	0.071614 41s	0.070027 27s
$trainscg$	0.071095 12s	0.072847 18s	0.069315 13s	0.069076 30s	0.071909 31s	0.072535 21s

4.1.4. Training algorithms. The goal behind introducing different training algorithms is to generate sufficiently accurate outputs. The weights and bias need to be updated during the training phase to obtain a function which performs correctly for a given input. Different training algorithms are discussed in [48]. Those training

algorithms are provided within *nprtool* in MATLAB environment. However, studying each algorithm deeply, in terms of definitions and related implementations, are far from the scope of this work, but the way of updating the weights, bias, and learning rate are the main differences between them. Our ANN has a single input in form of FFT value, and two output layers with the following targets: the first target is a vector of 0s where there is no voltage collapse, and 1s where there is a voltage collapse occurring; the second target vector is the complement of the first first target vector. The dataset used to train the neural network is divided as follows: 70% is selected randomly for the training phase, 15% is selected randomly for the validation, and the last 15% is selected for the testing.

To create sufficiently accurate and suitable output and architecture for our battery voltage collapse function, the obtained data from the battery model and the FFT process is tested with all training functions listed in the first column of Table 8. Each training function is tested with a different number of neurons. Mean squared error (MSE) and training time are reported in the mentioned table. Both of *trainlm* and *trainbr* show the best behavior in terms of MSE. However, *trainlm* has less training time comparing to *trainbr*, where *trainlm* confusion matrix and its performance are shown in Fig. 18 and Fig. 19, respectively. Note that, the n is selected as 60 neurons, and the overall accuracy is reported as 93.4%. Now, the ANN function needs to be used within an algorithm to do the stacking and FFT process in order to feed the ANN by the evaluated observations.

Basically, Algorithm 1 begins as follows: loop counter β is set to one, and battery selector S is set to one as well since the battery has full lifetime initially. Taking this into consideration, the outputs of ANN function are bounded between [0,1], where the 1^{st} ANN output produces *one* when that battery is operated in the unstable region, and it outputs *zero* at the stable region. As discussed previously, the 2^{nd} ANN output is the complement of the 1^{st} output. A predefined threshold γ is used to announce battery voltage collapse at early stages because the target vector covers early and late collapse observations, where the γ is placed on the 1^{st} ANN output only. Also, Upper and lower window limits α_1 and α_2 of N are needed. Initially, α_1 is set to one, while α_2 is set to window size value N. The algorithm needs to wait until N voltage measurements

are sampled. Then, FFT will be applied for the stacked N voltage measurements, $|f|$ for selected f_s will be collected, and $|f|$ is fed to ANN function for battery terminal voltage collapse checking purposes. The battery will be disconnected in case the 1^{st} ANN output layer is greater than γ. In case of keeping the battery connected to the load, α_1 and α_2 are incremented by $(N-1)$, then, the algorithm will wait again until the battery terminal voltage measurements are stacked within the new window to repeat the same checking process.

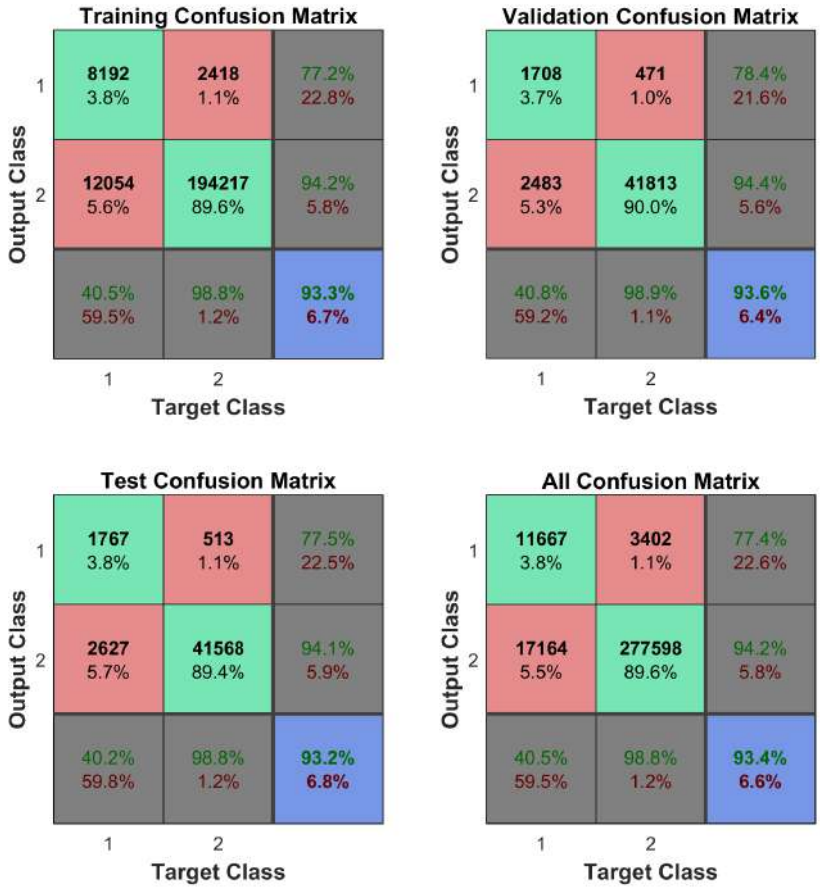

Figure 18: Confusion matrix.

Now, the ANN function is ready to detect an impending battery terminal voltage. In the next sections, the obtained ANN decision function will be investigated further in terms of detection performance during different actual tests. In addition, different

battery terminal voltage collapse detection methods will take place in this work for comparison purposes to end up with fair say about the proposed approach in this work.

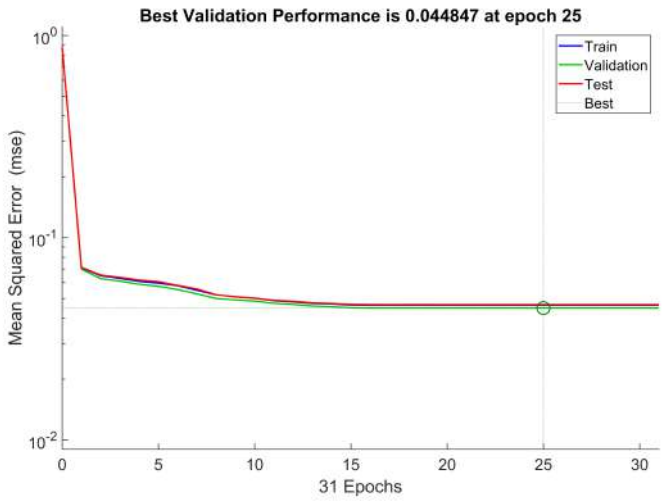

Figure 19: MSE for the training, validation, and testing data.

4.1.5. Preliminary results of battery terminal voltage collapse. In this section, the proposed method to detect battery terminal voltage collapse based on the ANN function is presented. The current and terminal voltage measurements are recorded via dSPACE (CP1104). In test1, the Li-ion battery is discharged with squared wave seen in Fig. 20, by switching the load bulb ON and OFF, until the battery is fully discharged. Further, the window size N is selected as 200 samples. Terminal voltage y samples are stacked and fed to FFT process to find out the frequency magnitude $|f|$ at f_s = 0.78125Hz for each window. By applying VT at 22V results to FA with *SOC* equal to 12.25%, while placing a CT at 10%, *SOC* was able to announce battery terminal voltage collapse correctly. The trained ANN also shows good performance in detecting battery voltage collapse correctly with *SOC* equal to 7.73% *SOC*. In test2, Li-ion battery got discharged continuously until battery terminal voltage collapse occurs as seen in Fig. 21. FA was the result of applying VT at 22V since the battery *SOC* approximately is equal to 17%. However, placing CT at 10% was capable of recognizing battery terminal voltage collapse accurately. The ANN announces a TA with battery *SOC* approximately equal to 10%.

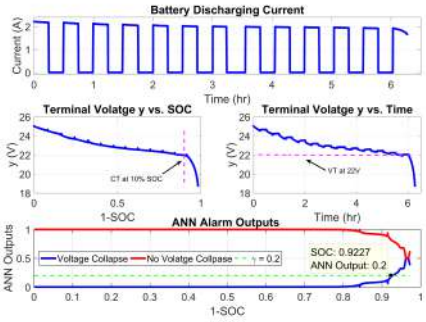

Figure 20: Test1, Li-ion battery is discharged with square wave load.

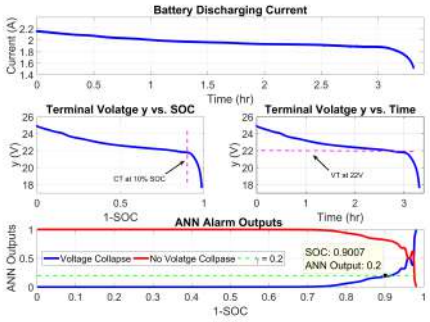

Figure 21: Test2, Li-ion battery is discharged continuously.

Algorithm 1 ANN battery terminal voltage collapse detection

Set: Loop counter β=1, battery selector S=1, predefined threshold $\gamma \in [0,1]$, window lower limit α_1=1, and window upper limit α_2= window size N
while $S == 1$ **do**
    Connect battery to the load
    $y(\beta)$ = read battery terminal voltage
    if *(β Modulus $N == 0$)* **then**
        Find FFT for $y(\alpha_1{:}\alpha_2)$
        Collect $|f|$ at selected f_s of $y(\alpha_1{:}\alpha_2)$
        Check the collected $|f|$ using the ANN function
        if *(ANN 1^{st} Output $>= \gamma$)* **then**
            Set: $S = 0$
        else
            Continue
        end if
        $\alpha_1 = \alpha_1 + (N-1)$
        $\alpha_2 = \alpha_2 + (N-1)$
    else
        Continue
    end if
    $\beta = \beta + 1$
end while

4.2. Motor Identification

Motor identification procedures in section 2.3 are applied in this section for the used motors in this work; the first motor is operated within dSPACE setup to show

initial results, while the second type is used with the mobile robot platform. In first stage, armature voltage (v_a), angular speed (ω), and armature current (i_a) at no-load steady-state condition are found, and the results are listed in Table 11.

4.2.1. Parameters of electrical equation. The back EMF constant (K_b) and torque constant (K_t) are obtained based on Eq. (15). As seen in Fig. 99, the linear fitting is applied for the back EMF (e_a) vs. angular speed (ω_{ss}), and then the back EMF constant (K_b) and torque constant (K_t) are found with value equal to 0.06578. In Table 12, motor resistance (R_a) is obtained under locked-rotor condition and same slope procedures are applied to evaluate R_a based on Eq. (16) with value equal to 0.6536 Ohm. The inductance of the motor (L_a) can be determined based on Eq. (17). Initially, the motor armature current is seen in Fig. 100. Further, the current (i_a) value before applying the voltage across the motor is found $\approx$ zero, while the i_a value after applying the voltage is 0.03937A. On the other hand, the voltage at applying moment is collected as seen in Fig. 101, as a result, the inductance (L_a) is found as 0.0363H.

4.2.2. Parameters of mechanical equation. The introduced Eq. (18) is applied to obtain the viscous friction (B_m), where the motor angular speed and current under no-load steady-state condition are seen in Table 11. The Eq. (18) is manipulated as given in Eq. (77), the motor angular speed vs. current is plotted to get the slope based on linear fitting function as seen in Fig (102), where the slope represents $\frac{K_t}{B_m}$, K_t is known ,and then the B_m is found equal to 2.4945×10^{-4}. The equivalent inertia (J_{eq}) is determined by running the motor in constant speed in the first stage, the power is switched off, at this moment, the J_{eq} is found equal to 5.2574×10^{-4} based on Eq. (19).

$$\omega_m = \frac{K_t}{B_m} i_a \tag{77}$$

4.2.3. Motor process reaction curve. The obtained motor parameters listed in Table 9 are used in simulation for validation purposes. process reaction curve (system response) is shown in Fig. 22, where the simulated system response based on the obtained parameters was unsuitable, therefore, the model is tuned by modifying mo-

tor resistance value, where both of obtained and modified system response are seen in Fig.22. Finally, GPC will be implemented in next section based on the modified motor parameters.

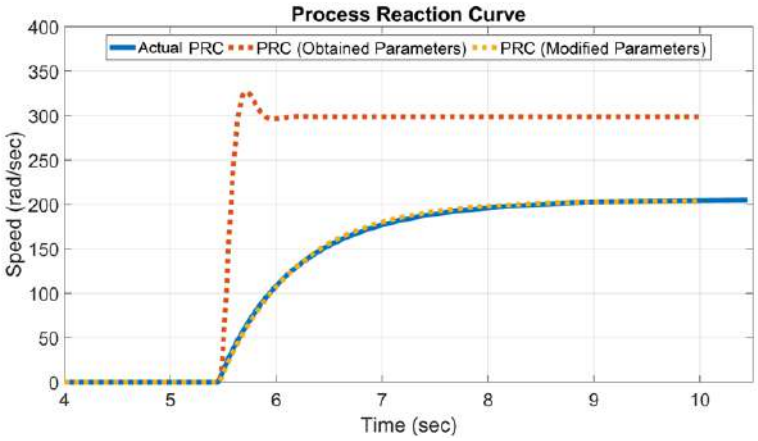

Figure 22: Motor process reaction curve

Table 9: dSPACE PMDC motor parameters

Parameter	Symbol	Obtained Parameter	Modified Parameter	Unit
Resistance	R_a	0.6538	9	Ohm
Back EMF constant	K_b	0.06578	0.06578	Nm/rad/s
Torque constant	K_t	0.06578	0.06578	Nm/A
Inductance	L_a	0.0363	0.0363	H
Viscous friction	B_m	2.4945×10^{-4}	2.4945×10^{-4}	
Equivalent inertia	J_{eq}	5.2574×10^{-4}	5.2574×10^{-4}	

4.3. Generalized Predictive Control (GPC)

In this section, GPC is used as speed controller and compared with a PI controller tuned by Cohen-Coon and Ziegler-Nichols rules, found in [49]. The comparison is performed in terms of transient time, disturbance rejection, controller effort, and different error criteria.

4.3.1. Transfer function. As discussed in [50], the continuous PMDC motor transfer function is obtained using MATLAB $ss2tf$ function, based on the rule in Eq. (78), where the continuous transfer function is seen in Eq. (79).

$$G(s) = \mathbf{C_c}\Big(sI - \mathbf{A_c}\Big)^{-1}\mathbf{B_c} \tag{78}$$

$$G(s) = \frac{\omega_m}{v_a} = \frac{K_t}{(L_aJ_{eq})s^2 + (R_aJ_{eq} + L_aB_m)s + R_aB_m + \dfrac{K_b}{K_t}} \tag{79}$$

Further, modified motor parameters in Table 9 are substituted into continuous transfer function, and it is then discretized based on *tustin* method with time step (Δ) equal to 0.01 sec. Note that the discretization is done using MATLAB $c2d$ function discussed in [51], where the final discrete transfer function is given in Eq. (80).

$$G(z) = \frac{\omega_m}{v_a} = \frac{b_1 + b_2z^{-1} + b_3z^{-2}}{a_1 + a_2z^{-1} + a_3z^{-2}} = \frac{0.9502 + 1.9z^{-1} + 0.9502z^{-2}}{1 - 0.781z^{-1} - 0.1968z^{-2}} \tag{80}$$

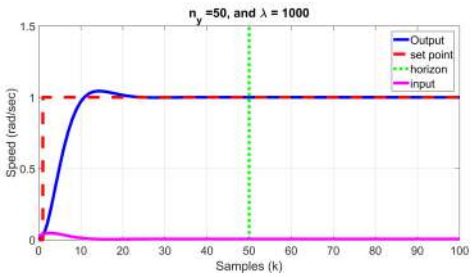

Figure 23: GPC stable response with n_y=50 and λ=1000.

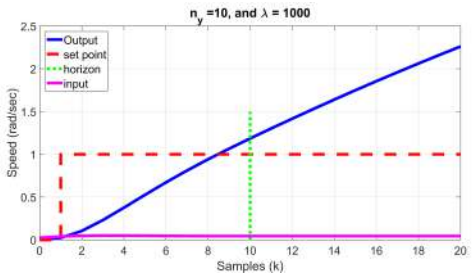

Figure 24: GPC unstable response with n_y=10 and λ=1000.

4.3.2. GPC tuning. The work discussed in section 2.4 is applied based on the obtained discrete transfer function in Eq. (80). Furthermore, GPC control law matrices (P_r, D, N) are built by selecting n_y and λ. As discussed in [37], a prediction horizon (n_y) needs to be greater than the settling-time of the used plant. In Fig. 23, simulated GPC stable response is shown based on n_y and λ equal to 50 samples and 1000, respectively. However, when the prediction horizon (n_y) is selected as 10 samples in case of λ = 1000, GPC shows unstable response as seen in Fig. 24. On the other hand, the stability

issue is fixed by changing λ from 1000 to 1 as seen in Fig. 25 and Fig. 26 since the λ affects the motor transient-time (τ_s).

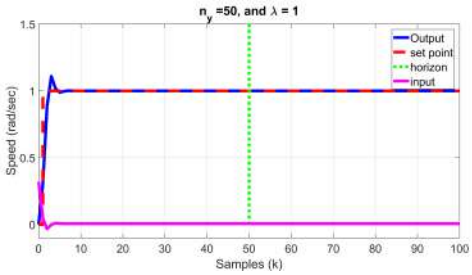

Figure 25: GPC stable response with n_y=50 and λ=1.

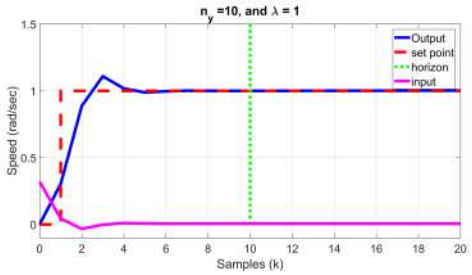

Figure 26: GPC stable response with n_y=10 and λ=1.

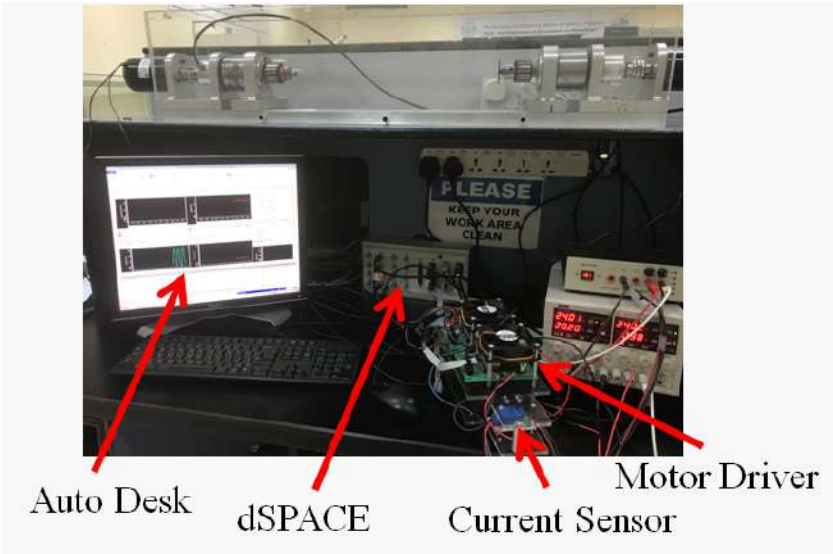

Figure 27: dSPACE setup.

4.3.3. Preliminary actual GPC step response results. Initially, GPC is constructed in MATLAB/Simulink and dSPACE environment as seen in Fig. 27, Fig. 104 and Fig. 105 to test controller response with respect to a step input. As illustrated in Fig. 28 and Fig. 29, GPC shows better settling-time and disturbance rejection comparing to PI (Cohen-Coon) and PID(Ziegler-Nichols), where n_y and λ are selected as 10 and 1, respectively.

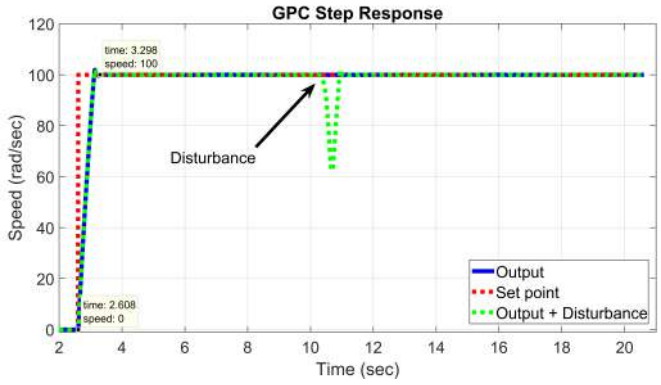

Figure 28: Step response test, GPC

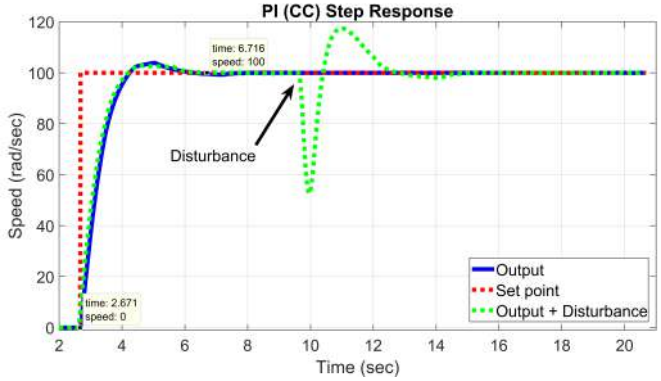

Figure 29: Step response test, PI (Cohen-Coon)

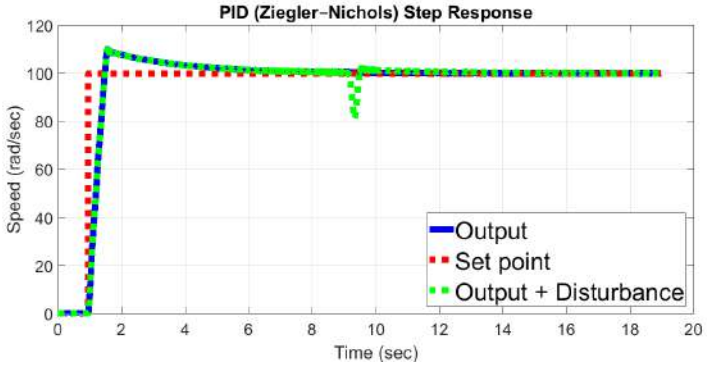

Figure 30: Step response test, PID (Ziegler–Nichols)

Theoretically speaking, integral squared error (ISE) penalizes huge errors more than smaller ones (due to squaring a big quantity ends with much bigger value), where

the ISE is used in this work to show large errors quickly. However, integral time-weighted absolute error (ITAE) penalizes the error after a long time much more than those at the beginning of the plant response. ITAE is a good observation tool to show the error occurred after the system settling-time. Different from ISE and ITAE, integrates absolute error (IAE) doesn't weight the error of plant response.

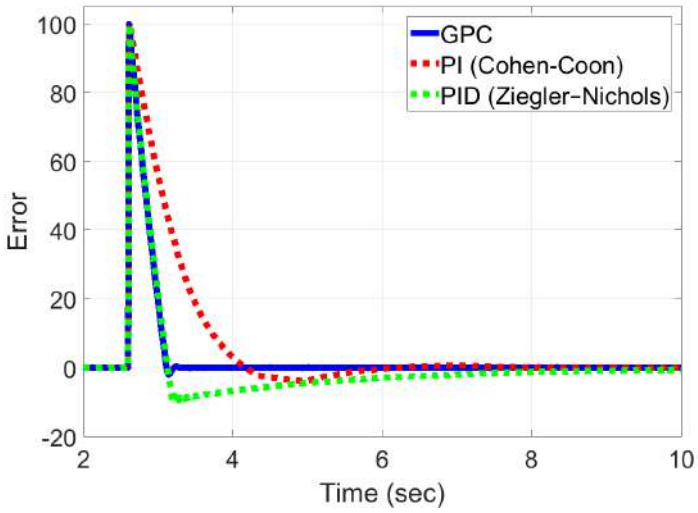

Figure 31: Step response test, Controllers error.

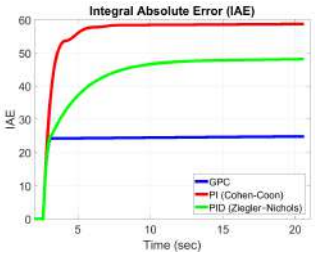

Figure 32: IAE.

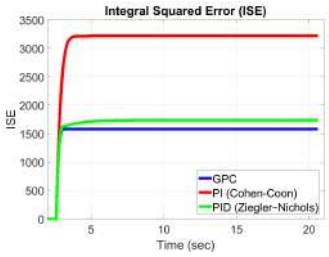

Figure 33: ISE.

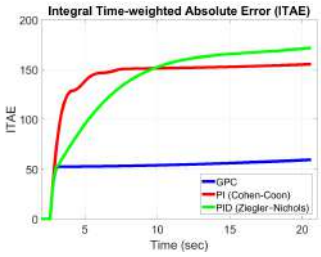

Figure 34: ITAE.

The error for each controller is shown by Fig. 31, where the GPC shows better performance in terms of driving the error to zero. Furthermore, different error criteria (IAE, ISE, and ITAE) are shown in Fig. 32, Fig. 33, and Fig. 34 where the GPC scores less amount of error in short and long term, compared to PI (Cohen-Coon) and

PID(Ziegler-Nichols). In terms of controller effort, GPC has a greater current consumption in short term. However, it reaches the steady-state value shortly as seen in Fig. 35.

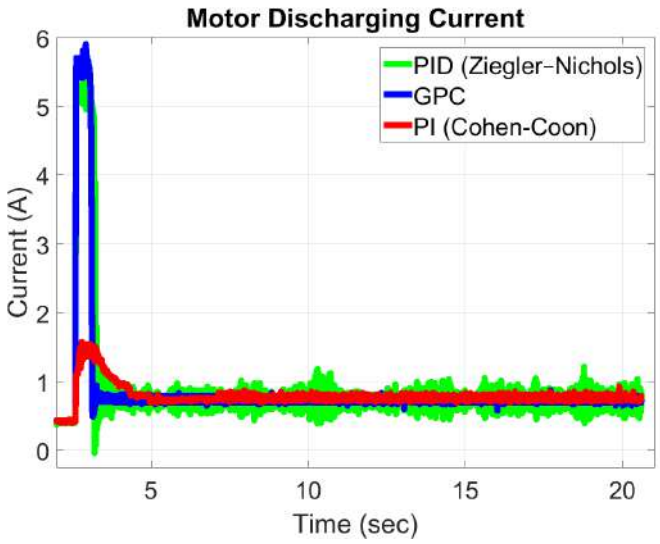

Figure 35: Step response test, controllers effort

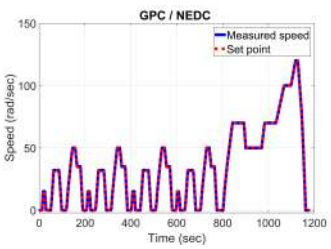

Figure 36: GPC.

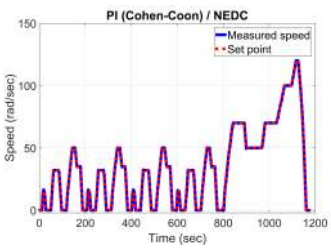

Figure 37: PI(CC).

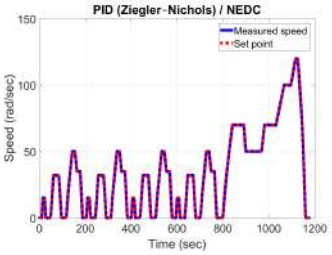

Figure 38: PID(ZN).

4.3.4. Preliminary actual GPC drive cycle results. In this test, a New European Driving Cycle (NEDC) is used as a referenced speed to the both controllers in order to test performance and effort with varying set points. As seen in Fig. 36, Fig. 37, and Fig. 38, controllers were able to follow the NEDC set points. Based on Fig. 40 and Fig. 41, GPC has good performance, compared to PI (Cohen-Coon), in terms of short and long term error, based on ISE and ITAE. However, IAE shows the absolute error without weighting it over the time as seen in Fig. 39. On the other hand, the PID (Ziegler-Nichols) has close error profile to GPC.

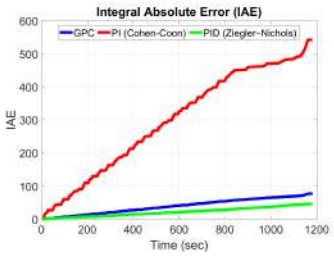

Figure 39: NEDC, IAE

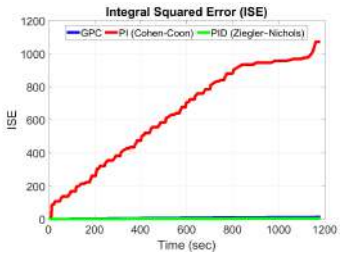

Figure 40: NEDC, ISE

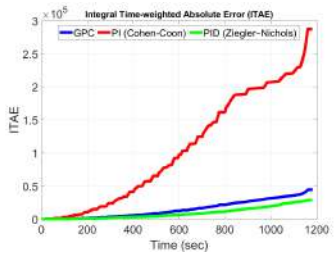

Figure 41: NEDC, ITAE

In Fig. 42, motor current measurement is seen, where PID (Ziegler-Nichols) has higher effort than GPC and PI (Cohen-Coon). However, GPC has clean current profile compared to PI (Cohen-Coon). As a result, GPC shows reasonable effort, performance, error, and transient time compared to PI (Cohen-Coon) and PID (Ziegler-Nichols). Furthermore, Applying GPC as robot low level controller will lead to discharge the stored power within the sources efficiently.

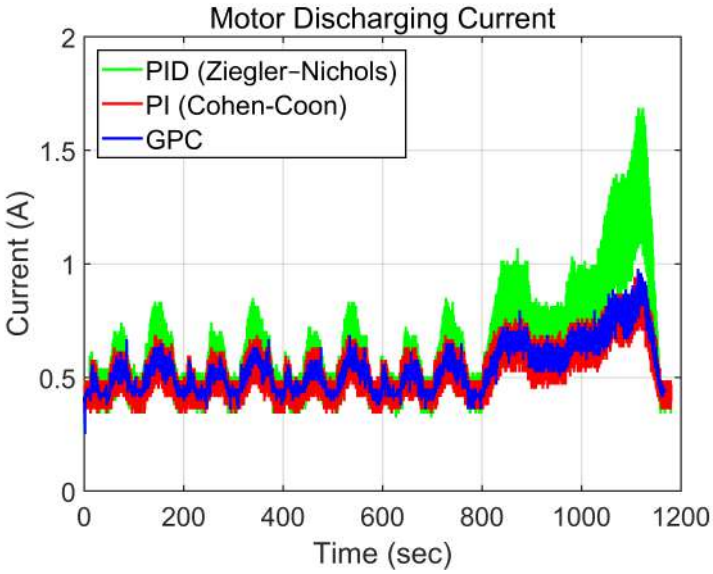

Figure 42: NEDC, Controllers effort.

4.4. Robot Overall System

As seen in Fig. 43, robot power system consists of a fuel cell (FC), two batteries, and a supercapacitor. Power sources are also connected in the form of passive parallel connection, where the main aim is to give the sufficient flexibility to the power system

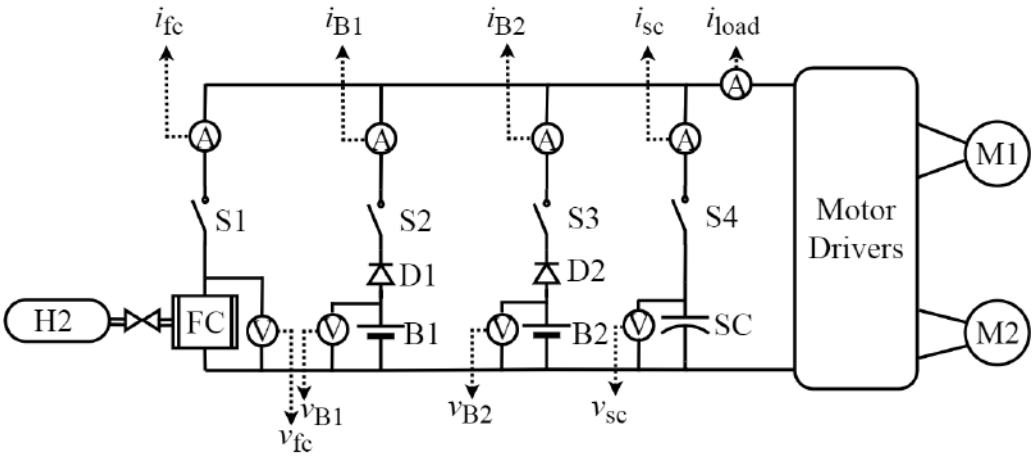

Figure 43: Power sources connection diagram

since the sources can be connected/disconnected independently from each other, where an electric switches (S_1, S_2, S_3, and S_4) are placed to do the required selection. The two batteries are protected from flow back current using diodes (D_1, and D_2). Different from all used power sources, FC is combined with hydrogen (H_2) cylinder containing the fuel.

Furthermore, voltage and current sensors are combined with each source to collect the required measurements, where the measurements are fed to a laptop placed on top of the robot. The sensors are interfaced first to a Arduino board, and then MATLAB is used in real-time to read sensors and control with help of a package discussed in [52], where the mentioned package converts MATLAB script to C-code. Initially, the process starts with connecting the selected sources by the user, where different tests are performed based on battery, battery-supercapacitor, and battery-supercapacitor-fuel cell.

After selecting the power sources combination, the error between the desired position (x_{des}, y_{des}) and robot initial position is found, where the robot localize itself initially based on LLA using MIDG-II INS/GPS sensor, and then it converts the position data from LLA to xyz in global frame based on Eq.(70), Eq.(71), and Eq.(72). The position in robot frame (RF) is found based Eq. (73). High level control (IOSFL) calculates v and ω based on the obtained error between the desired and measured position, where the robot orientation (θ) is determined based on Eq. (54). Furthermore, the obtained v and ω are converted to controller set points (reference) in form of right (v_R)

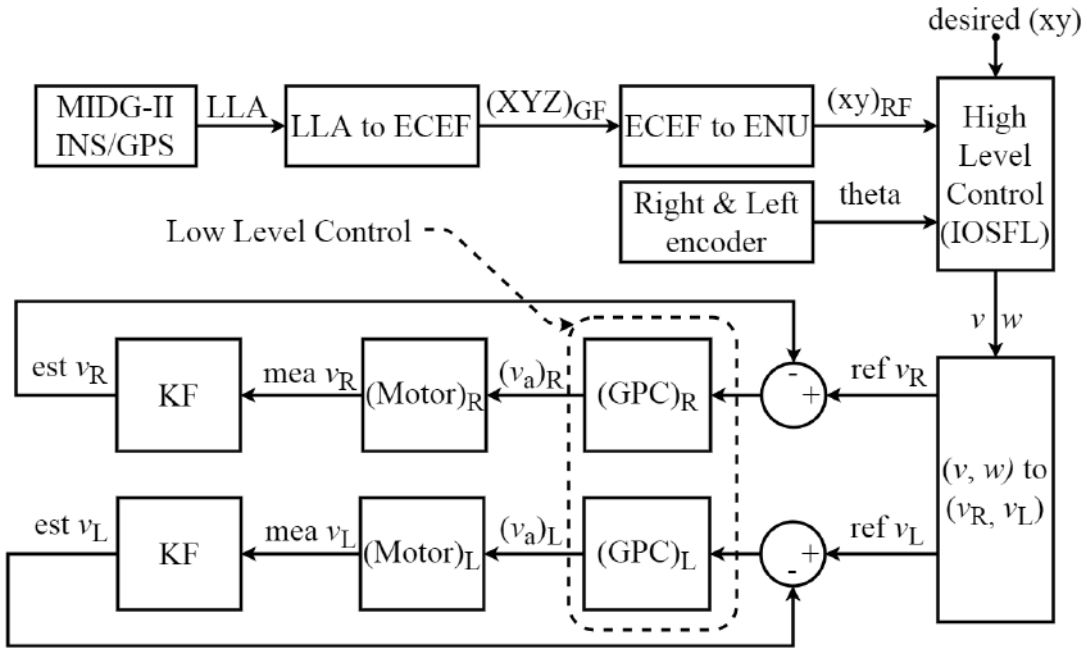

Figure 44: High and low level control diagram

and left (v_L) speeds, where Eq. (58) and Eq. (59) are used to accomplish the conversion. The measured speed from each motor is filtered by Kalman filter (KF) to reduce the noise effect before feeding the estimated speed back to the error node as seen in Fig. 44. The robot inside layer is shown in Fig. 45, where it consists of PMDC motors, incremental encoders, motor drivers, relays, and common ground. On the other hand, the outside layer is seen in Fig. 46, where it contains the laptop, MIDG-II INS/GPS sensor, batteries, fuel cell, hydrogen cylinder, supercapacitor, current sensors, and voltage sensors. In next chapter both of battery voltage collapse method and GPC controller will be investigated more using a differential drive robot.

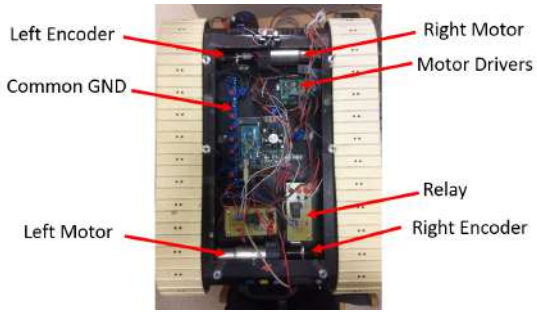

Figure 45: Robot inside layer

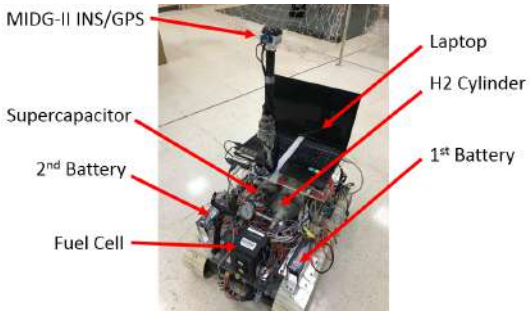

Figure 46: Robot outside layer

Chapter 5: Field Test Results and Analysis

In this chapter, the proposed method in section (4.1) to detect battery terminal voltage collapse based on FFT and ANN function is presented, where the preliminary tests are shown in section (4.1.5) for the Li-ion battery. In addition, the discussed high level control (IOSFL), and low level control in form of GPC combined with KF in section (4.4) are tested with differential-drive robot.

5.1. KF Effect on GPC Performance

As discussed in section (2.4), the GPC has a filtering term in the form of moving average $T(z)$. As discussed previously, $T(z)$ is equated to 1, and then it is replaced by KF to reduce the noise effect. In this section, GPC performance will examine with and without KF algorithm discussed in section (2.5). The control law given by Eq. (44) represents the GPC in case of excluding the KF, while the control law in Eq. (81) describes GPC in case of including the KF, where $\underleftarrow{y}_k$ is replaced by $\underleftarrow{\hat{y}}_k$.

$$\underrightarrow{\Delta u}_k = P_r \underrightarrow{r}_k - D\underleftarrow{\Delta u}_{k-1} - N\underleftarrow{\hat{y}}_k \tag{81}$$

KF algorithm in section (2.5), PMDC discrete model in section (2.2), robot PMDC motor parameters listed in Table 13, GPC algorithm in section (2.4), and high level control (IOSFL) discussed in section (2.7) are used to accomplish this task effectively. Furthermore, the process and measurement noise covariance of KF are selected as 0.03 and 0.1, respectively. Note that, the values are obtained by trial and error.

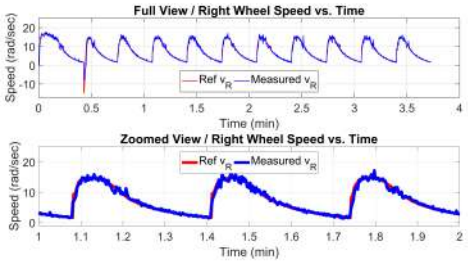

Figure 47: Reference and measured speed of the right wheel.

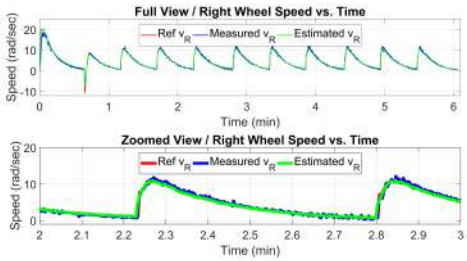

Figure 48: Reference, measured, estimated speed of the right wheel.

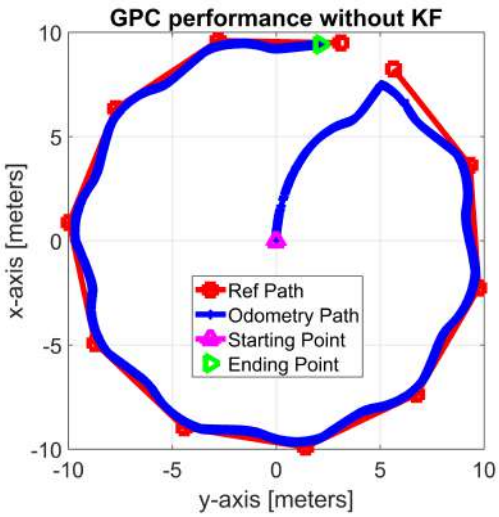

Figure 49: Reference and robot path without including KF.

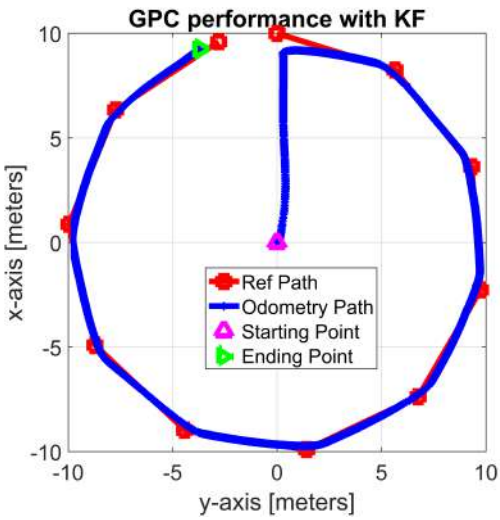

Figure 50: Reference and robot path including KF.

As discussed in [37], one of the weak points of a GPC controller is its sensitivity to the noisy measurements. Therefore, the shown speed measurement combined with noise in Fig. 47 is fed to the GPC to study its performance. However, the filtered measurement seen in Fig. 48 is used within GPC in this second case. Furthermore, the same work is applied to the left wheel. Clearly from Fig. 49 and Fig. 50, including the KF leads to enhancing robot performance in terms of path following. Since the shortest path between two points is a straight-line, the GPC without KF is going to draw more power from the sources with time compared to a GPC combined with KF. As a result, GPC performance powered by different sources will be investigated more in next section.

5.2. GPC Effort with Different Sources

In this section, the differential drive robot discussed in section (4.4) is used to test GPC performance combined with KF, where the GPC preliminary results are seen in section (4.3.3). Furthermore, different tests are performed in case of using a single battery, battery-supercapacitor, and battery-supercapacitor-fuel cell to run the robot along predefined path.

5.2.1. Battery test. In this test, single Li-ion battery is used to supply robot power demand while it is moves along the seen path in Fig. 51 and Fig. 52. In Fig.

51, GPS visualizer site is used to plot the recorded LLA data during the test on Google map. Equations discussed in section (2.8.1) and section (2.8.2) are applied to obtain the GPS path seen in Fig. 52 since the target is to localize the robot in xyz–coordinates. The obtained GPS path is compared with Odometry path discussed in section (2.6.1).

Figure 51: Battery test, LLA robot path on Google map using GPS visualizer.

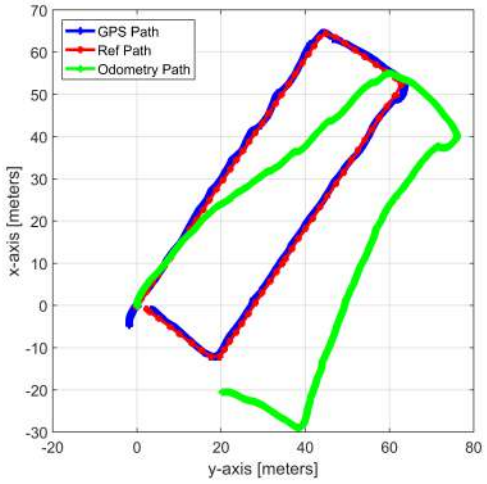

Figure 52: Battery test, reference and robot path

In Fig. 52, robot GPS path in ENU format is used as feedback to the high level control (IOSFL) in order to drive the robot along the reference path. However, the Odometery path was inaccurate during the test since no measurement is prefect and model states $[x \;\; y \;\; \theta]^T$ are obtained based on integrating v_R, v_L, and θ combined with noise.

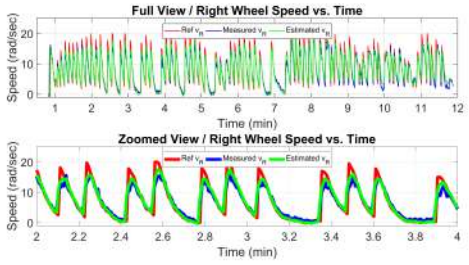

Figure 53: Battery test, reference, measured, estimated speed of the right wheel.

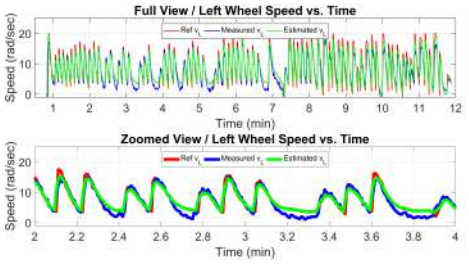

Figure 54: Battery test, reference, measured, estimated speed of the left wheel.

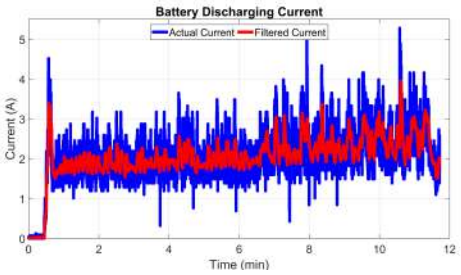

Figure 55: Battery test, discharging current.

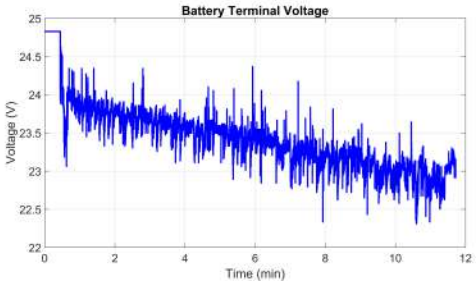

Figure 56: Battery test, battery terminal voltage.

The right (v_R) and left (v_L) wheel speed are seen in Fig. 53 and Fig. 54, where the estimated speed is fed to GPC during the test to enhance its accuracy, as discussed in section (5.1). Clearly, the estimated speed over/under fit the measurement instead of smoothing it in some places, while process and measurement noise covariance selected values are causing this behavior. The work found in [53] introduces a method to compute process and measurement noise covariance in each time step such that KF works as optimal state estimator. On the other hand, battery terminal voltage and discharging current are seen in Fig. 55 and Fig. 56, respectively. The drop in battery state of charge (SOC) will be discussed later after finalizing all tests.

5.2.2. Battery–supercapacitor test. In this test, same procedures in terms of high level (IOSFL) and low level (GPC) control discussed previously in section (5.2.1) are applied. Furthermore, the recorded LLA and robot path during battery-supercapacitor test are seen in Fig. 57 and Fig. 58, respectively. Different from battery test in section (5.2.1), the supercapacitor is connected in parallel connection with the battery during this experiment to study GPC performance.

As discussed in [5], a supercapacitor can capture the produced energy by motor in regenerative braking period, where the work aims to enhancing battery lifetime with the help of power controller. Theoretically speaking, a power controller is needed to let the the supercapactour charge during the regenerative braking period and discharge with respect to certain conditions, where DC/DC converter controls the flow of the energy. In this work, the supercapacitor is connected to the battery directly without including a power controller or DC/DC converter in between to avoid weight issues.

Figure 57: Battery-supercapacitor test, LLA robot path on Google map using GPS visualizer.

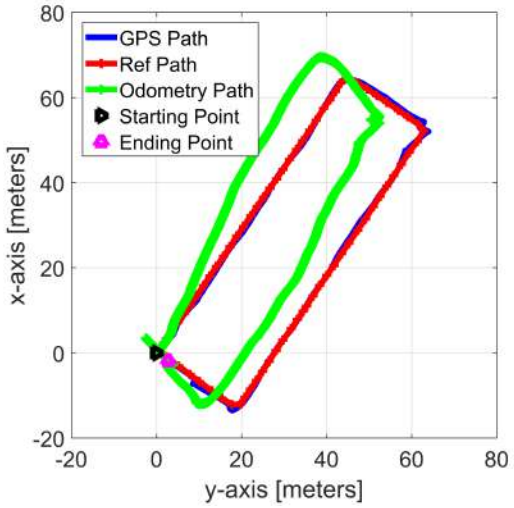

Figure 58: Battery-supercapacitor test, reference and robot path

The zoomed view of right (v_R) and left (v_L) wheel speed in Fig. 59 and Fig. 60 show the robot while it is trying to adjust its direction at the end of the trip; as a result, both of v_R and v_L are fluctuated between positive and negative speed values. Based on this fact, the nominal value of the battery discharging current seen in Fig. 61 gets decreased because of the energy flow from the motor back to supercapacitor, where the supercapacitor charging current at that moment is shown in Fig. 62.

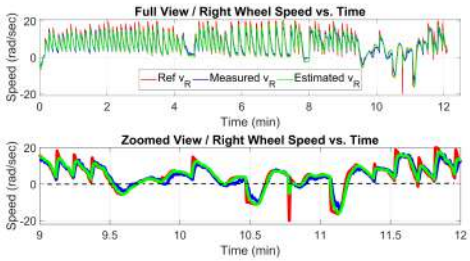

Figure 59: Battery-supercapacitor test, reference, measured, estimated speed of the right wheel.

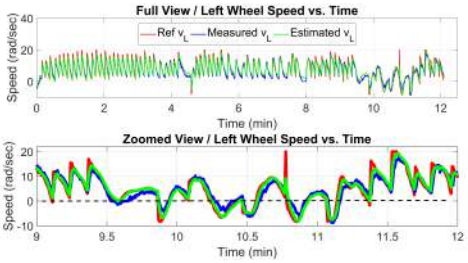

Figure 60: Battery-supercapacitor test, reference, measured, estimated speed of the left wheel.

Supercapacitor covers load demand first for short time and then the battery supplies the remaining power to the load since a supercapacitor has fast response, compared to battery. As seen in Fig. 62, supercapacitor is kept charging/discharging since it pro-

vides the energy first, and then it gets charged by the battery when its voltage becomes less than battery voltage. Therefore, keeping the supercapacitor connected to the load affects battery lifetime badly. However, letting supercapacitor connected to the load at the right moment can enhance battery lifetime since it can capture the flow back energy with the help of a power controller and DC/DC converter as discussed previously.

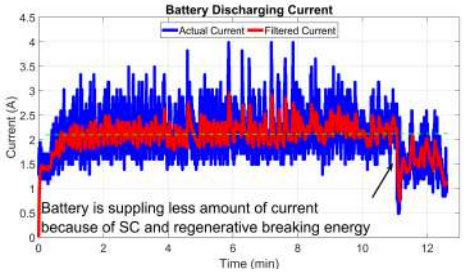

Figure 61: Battery-supercapacitor test, battery discharging current.

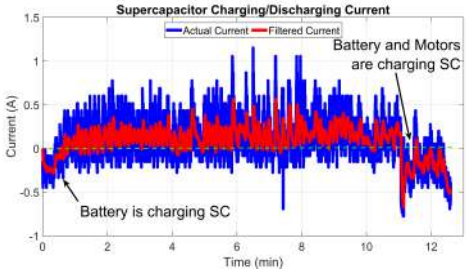

Figure 62: Battery-supercapacitor test, supercapacitor charging/discharging current.

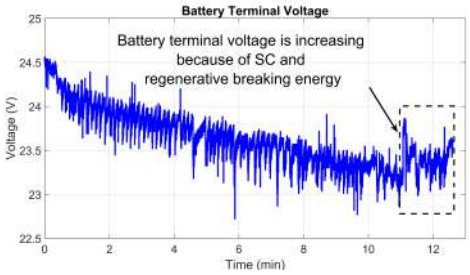

Figure 63: Battery-supercapacitor test, battery terminal voltage.

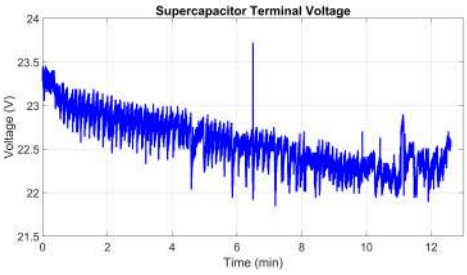

Figure 64: Battery-supercapacitor test, supercapacitor terminal voltage.

5.2.3. Battery–fuel cell test. In this test, a fuel cell is connected to the battery in parallel configuration. Furthermore, same previous tests procedures are applied in terms of high and low level control. In Fig. 65, the LLA recorded data during the test is plotted on Google map using GPS visualizer site. However, robot path after converting the LLA to XYZ-coordinates and applying ENU is shown in Fig. 66, where the Odometry path shows the same poor behavior during the test.

Reference, measured, estimated speed for the right and left wheel are shown in Fig. 67 and Fig. 68, where fluctuations between positive and negative values are shown

in the zoomed view. However, the produced energy is not capture due the absence of the supercapacitor in this test.

Figure 65: Battery-fuel cell test, LLA robot path on Google map using GPS visualizer.

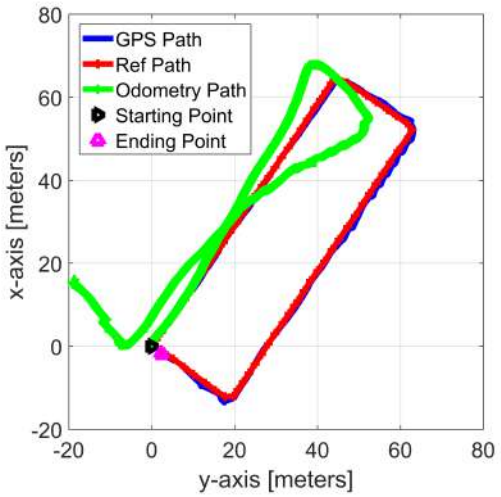

Figure 66: Battery-fuel cell test, reference and robot path

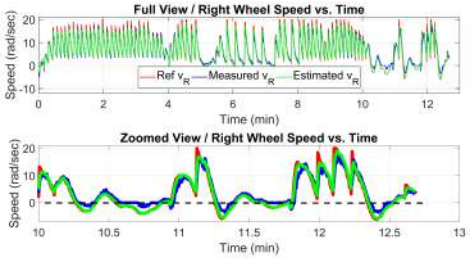

Figure 67: Battery-fuel cell test, reference, measured, estimated speed of the right wheel.

Figure 68: Battery-fuel cell test, reference, measured, estimated speed of the left wheel.

The main advantage of using fuel cell in this test is seen in Fig. 69 and Fig. 70 compared to previous tests, where the battery supplies less amount of current, and the fuel cell provides the remaining current demand since the fuel cell has higher terminal voltage as shown in Fig. 71 and Fig. 72. As a result, battery lifetime can be enhanced through sharing the load demand.

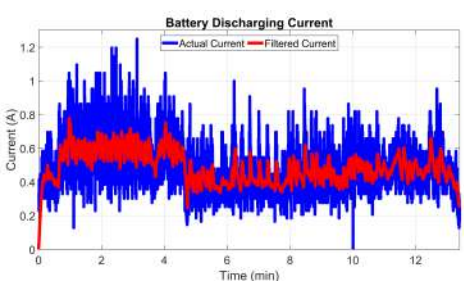

Figure 69: Battery-fuel cell test, battery discharging current.

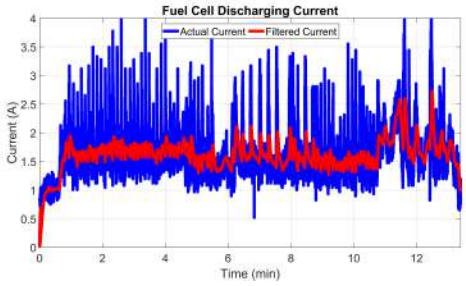

Figure 70: Battery-fuel cell test, fuel cell discharging current.

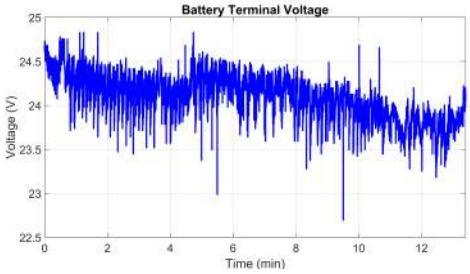

Figure 71: Battery-fuel cell test, battery terminal voltage.

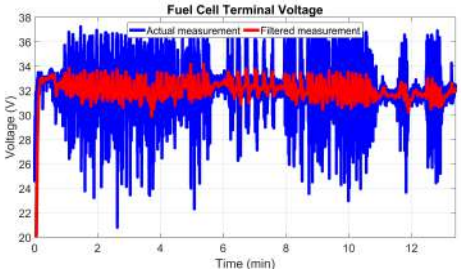

Figure 72: Battery-fuel cell test, fuel cell terminal voltage.

5.2.4. Battery–supercapacitor–fuel cell test. In this test, power sources combination consists of a fuel cell, supercapacitor, and battery, where the power sources are connected in parallel configuration. Also, same high and low level control are applied during this experiment. In Fig. 65, Google map and GPS visualizer site are used to show LLA recorded data during the test. As seen in Fig. 74, robot path in form of xyz–coordinates is obtained. Clearly, Odometry model shows inconstancy in terms of localization accuracy compared to GPS approach.

In Fig. 75 and Fig. 76, right and left wheel speed are shown for the full trip. In the zoomed view, both motors are fluctuated to adjust robot orientation towered the desired orientation, where the cause of the fluctuation is back to the noisy GPS measurement at that moment. As a result, the supercapacitor is charged by the produced energy at that moment as seen in Fig. 79, as well as, both of battery and fuel cell are supplied less amount of current under this condition. Furthermore, the same issue is

seen in battery, fuel cell, and supercapacitor terminal voltage in Fig. 80, Fig. 81, and Fig. 82, respectively.

Figure 73: Battery-supercapacitor-fuel cell test, LLA robot path on Google map using GPS visualizer.

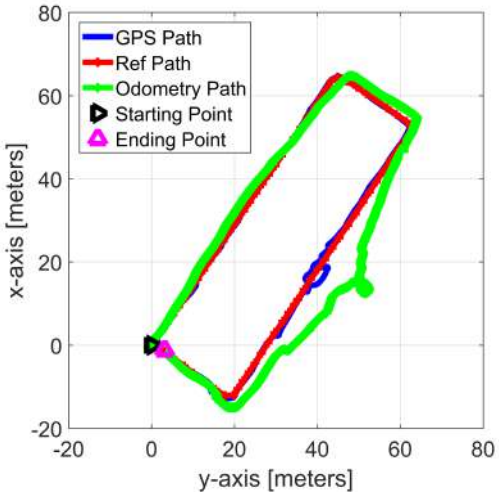

Figure 74: Battery-supercapacitor-fuel cell test, reference and robot path.

Figure 75: Battery-supercapacitor-fuel cell test, reference, measured, estimated speed of the right wheel.

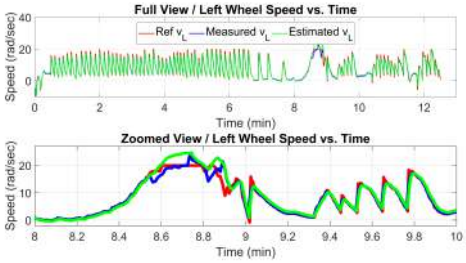

Figure 76: Battery-supercapacitor-fuel cell test, reference, measured, estimated speed of the left wheel.

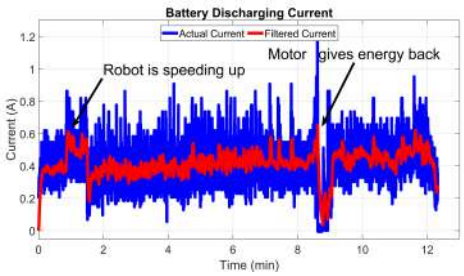

Figure 77: Battery-supercapacitor-fuel cell test, battery discharging current.

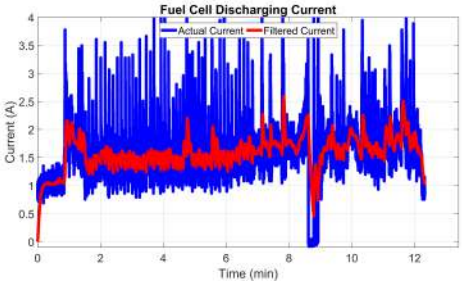

Figure 78: Battery-supercapacitor-fuel cell test, fuel cell discharging current.

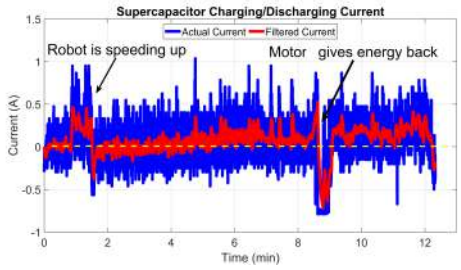

Figure 79: Battery-supercapacitor-fuel cell test, supercapacitor charging/discharging current.

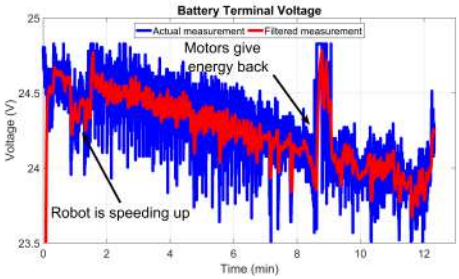

Figure 80: Battery-supercapacitor-fuel cell test, battery terminal voltage.

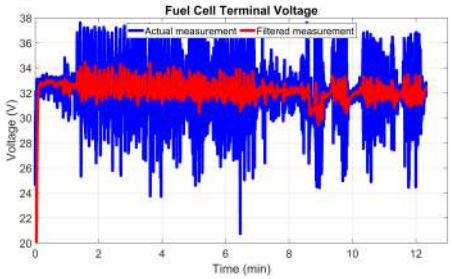

Figure 81: Battery-supercapacitor-fuel cell test, fuel cell terminal voltage.

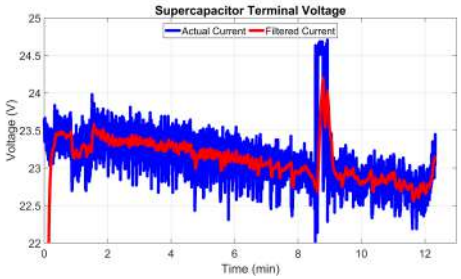

Figure 82: Battery-SC-FC test, supercapacitor terminal voltage.

5.2.5. Time analysis for different tests. Battery *SOC* is calculated for the previous tests using Coulomb-counting in discrete form given in Eq. (82), where i_k represents battery current measurement, C_c is battery capacity given in Eq. (1), Δ is the time-step, and the initial SOC_k is set to one since the battery has full lifetime in the beginning. As seen in Fig. 83, battery–supercapacitor configuration affects battery lifetime badly in long term since the battery is supplying the load demand and super-

capacitor. Although, adding the suppercapacitor gives the chance to recover energy flowed back from the motors., different from battery and battery–supercapacitor tests, battery–fuel cell configuration offers better battery lifetime since fuel cell covers most of the load demand. Furthermore, this type of conflagration does not have the ability to recover any energy from the motor. However, battery-supercapacitor-fuel cell combination shows the best performance in terms of battery lifetime, where the same issue is reported in [54].

$$SOC_{k+1} = SOC_k - \frac{i_k}{C_c}\Delta \tag{82}$$

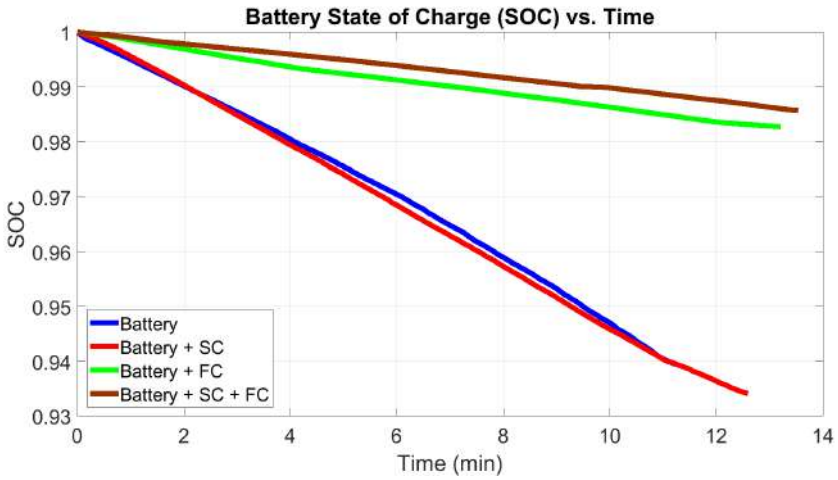

Figure 83: Battery lifetime for different tests

The full lifetime is calculated based on Eq. (83), where the idea is to show full drop in battery lifetime. Initial state of charge (SOC_1), final state of charge (SOC_2), initial time (T_1) are equated to 1, 0, 0 min, respectively. In addition, the slope for each test is found and listed in Table 10. As a result, the final battery lifetime for each test is calculated and listed in Table 10. As seen in Fig. 84, battery-supercapacitor-fuel cell has higher lifetime than other combinations since it could last to 16.65 hr. Further, combining the battery with fuel cell lets the battery last in operation for 12.804 hr. However, both battery and battery supercapacitor test have approximately same battery lifetime.

$$SOC_2 - SOC_1 = a(T_2 - T_1) \tag{83}$$

Table 10: Slope and final battery lifetime for each test.

Test Type	Slope (a)	T_2 (hr)
Battery	0.0054	3.069
Battery+SC	0.0052	3.1885
Battery+FC	0.0011	12.804
Battery+SC+FC	0.001	16.65

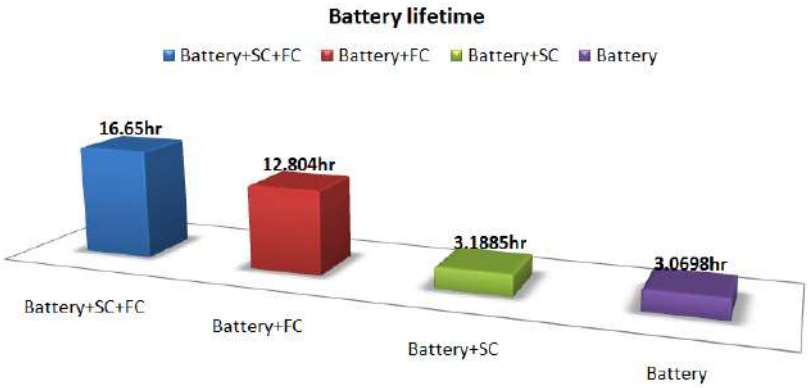

Figure 84: Final battery lifetime

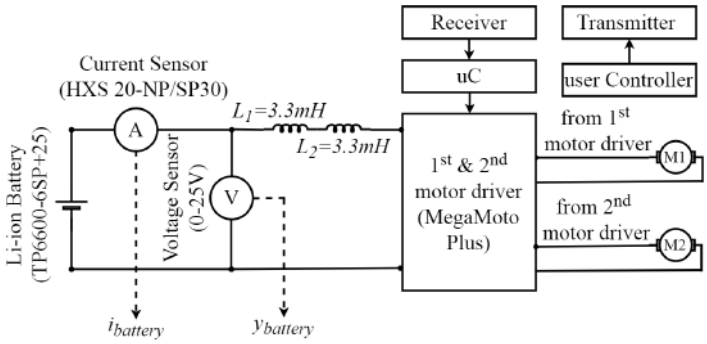

Figure 85: Robot connections diagram

5.3. Battery Terminal Voltage Collapse Detection based on FFT and ANN Function

5.3.1. ANN battery test. A differential-drive robot is used to discharge a full lifetime battery until the ANN announces battery voltage collapse, and ANN detection performance is compared by convectional techniques such as VT and CT. Note that, power sources (2^{nd} battery, fuel cell, supercapacitor) shown in Fig. 46 are not used in this test. In the preliminary tests, the voltage and current measurements were sufficient because the load was simple resistive load. However, the loads in this test are two DC motors, in addition to motor drivers. Practically speaking, those motor drives add more difficulties due their switching behavior. As seen in Fig. 85, two inductors (L=6.6 mH) are placed to filter the measured current. The motor drivers are used to control left and right motor speed via PWM signal generated by a micro-controller. A user controls the robot through wireless RC-transmitter ($Futaba-6J-2.4\ GHZ$). The receiver is fed to the micro-controller.

Figure 86: Longitude vs. latitude recorded via MIDG II INS/GPS sensor.

In this test, the robot is driven continuously along the seen path in Fig. 86 until the trained ANN detects battery voltage collapse, where the path is a part of the parking area in the American University of Sharjah (AUS). GPS data is recorded during the experiment via MIDG II INS/GPS sensor in real time during the test. The ANN output threshold value (γ) is selected equal to 0.5; however, ANN is performed batter with $\gamma = 0.2$ as realized later. The actual and filtered motors current are seen in Fig. 87, where a low pass filter with a cut-off frequency equal to 5Hz is used to show the average current value. Note that, the calculated SOC is inaccurate due to noisy current measurement; base on Fig. 87, applying CT announces LA. Regarding VT, the method leads to FA since the battery terminal voltage hit the 22V threshold at early stages. Different from VT and CT, the trained ANN is capable of declaring TA with $\gamma = 0.2$ as seen in Fig. 87, where the ANN output plotted vs. time (hr) instead of *SOC* due the calculated values are not suitable.

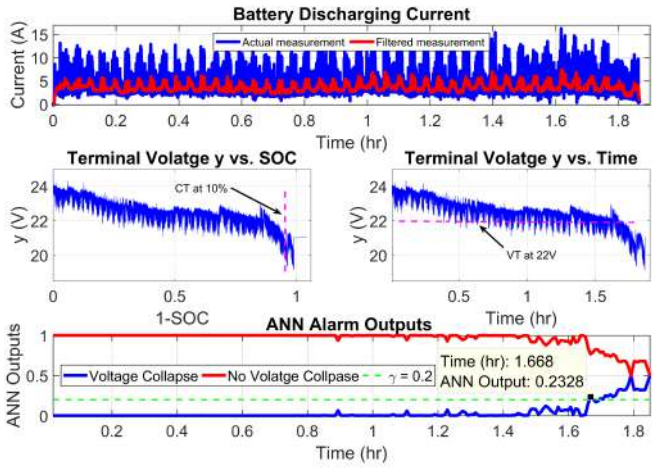

Figure 87: ANN battery test, ANN outputs vs. time

5.3.2. ANN battery–supercapacitor test. The differential-drive robot controlled by high and low level control is combined with ANN algorithm to detect battery terminal voltage collapse in real-time. A battery with low capacity is used since full capacity needs long time to discharge as seen in Fig. 87. Furthermore, LLA robot

path during the test is seen in Fig. 88, and GPS path compared to Odometry path after applying ENU is shown in Fig. 89.

In this test, ANN detection algorithm is used to replace the dying battery by backup one in real-time. The ANN alarm outputs are shown in Fig. 90, where the threshold (γ) is selected as 0.3. As seen in Fig. 90, the voltage collapse alarm signal is crossed the alarm threshold at time equal to 1.285 min. Based on that, the 1^{st} battery is terminated from the load and the 2^{nd} battery is connected, where 1^{st} battery terminal voltage processed within ANN algorithm is seen in Fig. 96. Furthermore, ANN algorithm does not affect by connecting a supercapacitor to the battery. Also, the low level controller (GPC) has not influenced by the battery switching since right and left wheel speed tracking are maintained.

Figure 88: ANN battery-supercapacitor test, LLA robot path is plotted on Google map using GPS visualizer.

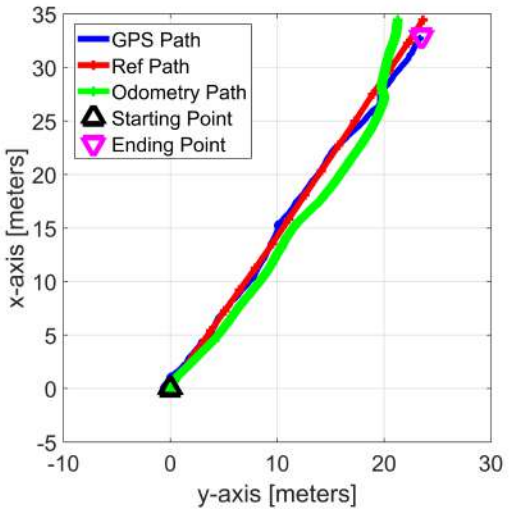

Figure 89: ANN battery-supercapacitor test, Reference and robot path

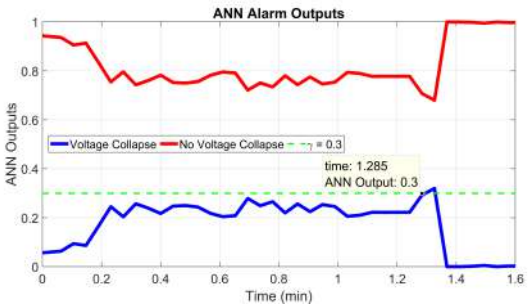

Figure 90: ANN battery-supercapacitor test, ANN outputs vs. time

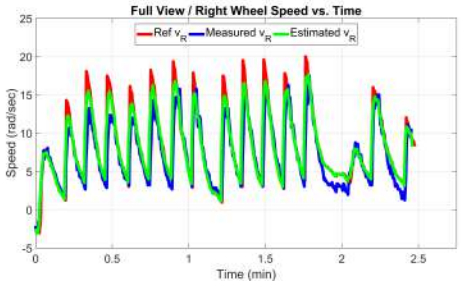

Figure 91: ANN battery-supercapacitor test, Reference, measured, estimated speed of the right wheel.

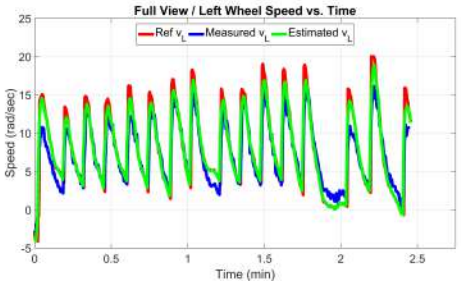

Figure 92: ANN battery-supercapacitor test, Reference, measured, estimated speed of the left wheel.

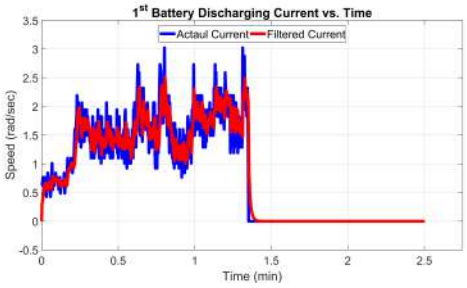

Figure 93: ANN battery-supercapacitor test, 1^{st} battery discharging current.

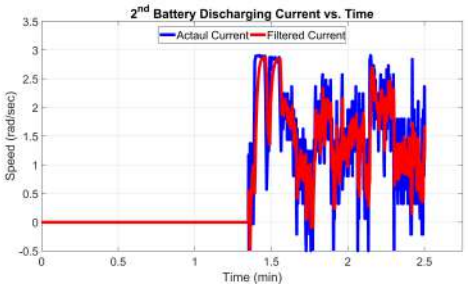

Figure 94: ANN battery-supercapacitor test, 2^{nd} battery discharging current.

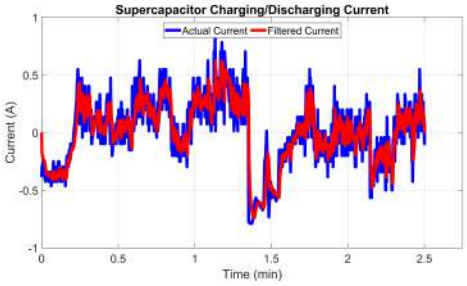

Figure 95: ANN battery-supercapacitor test, supercapacitor charging/discharging current.

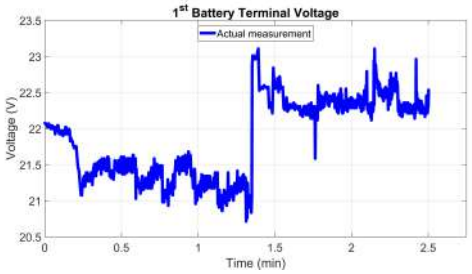

Figure 96: ANN battery-supercapacitor test, 1^{st} battery terminal voltage.

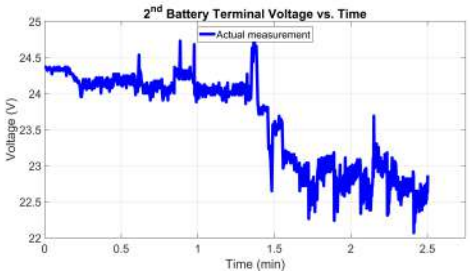

Figure 97: ANN battery-supercapacitor test, 2^{nd} terminal voltage.

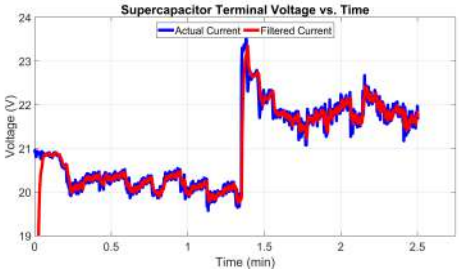

Figure 98: ANN battery-supercapacitor test, supercapacitor terminal voltage.

Chapter 6: Conclusion

A novel neural network method is introduced in this work to detect battery voltage collapse and alarm the end-user about the failure when it is going to happen. The method successfully predicted battery voltage collapse in real-time battery monitoring. The proposed method can be used in protecting Li-ion battery from an over-discharge scenario. According to Table 14, the trained ANN function scored three TAs out of three different tests. Where VT ends with zero TA and three FAs; however, VT was close to the 10% *SOC* in the preliminary test1 and 2 as listed in Table 14. CT shows better performance by scoring two TAs and one LAs, where one way to enhance CT performance is estimating *SOC* using KF and EKF [27]. However, this technique requires a battery model with sufficient accuracy. Linearization is required in case of using nonlinear battery model like Chen and Mora's [28]. In addition, EKF works as optimal state estimator when the process and measurement noise are statistically estimated as discussed in [53]. In [55], adaptive unscented Kalman filter associated with a noise statistics estimator is proposed to solve *SOC* accuracy calculation issues as well.

The proposed method has weak points in terms of the empirical way of defining the window size N, where the same empirical issue is applied for the pecked $|f|$ at selected frequency f_s, and predefined threshold γ which affects the accuracy of the method since the ANN is trained to output *one* for the early and late voltage collapse observation in terms of FFT value. Further, ANN requires extensive training based on accurate battery model with known parameters for the used battery in actual tests. Finlay, this method can be enhanced further by adding new observations to ANN like battery temperature and surface acoustic waves [56] since the multi-sensor approach offers more knowledge about battery terminal voltage transition from stable to the unstable region.

A generalized predictive control (GPC) combined with Kalman filter (KF) is used to control motor speed, where the main aim is to generate the minimum input voltage to the motors since the control law is obtained by minimizing a cost function with respect to Δu. However, the cost function provides a suboptimal solution since it is approximated with number of samples greater than system settling time. Preliminary controller tests, GPC shows good performance, compared to a PI controller tuned by Cohen-Coon rules in terms of transit time, disturbance rejection, and different error

criteria. In step response test, GPC draws more power than PI (Cohen-Coon) since it is accelerated from zero to 100 rad/sec in short time to drive the error to zero; however, PI (Cohen-Coon) consumes less amount of current but it takes longer time to drive the error to zero. Different from step response test, GPC performs better than PI (Cohen-Coon) in drive cycle test in terms the same comparing issues, where it is shows almost the same current profile since the reference speed is changing smoothly.

Different power sources, sensors, relays, motor drivers, GPS sensor, and laptop are mounted on differential-drive robot to test the ANN and GPC performance in real-time outdoor tests. An input/output state feedback linearization (IOSFL) used high level control to generate linear (v) and rotational robot velocity (ω), where the obtained v and ω are used to calculate right and left wheel reference speed to the GPC. A GPS sensor (MIDG II INS/GPS) is used to localizes robot position. Furthermore, robot location given by GPS sensor and desired points in each time step feds to the IOSFL to generate the correspond robot v and ω. The effect of KF on GPC performance is also discussed, where adding KF to GPC leads to improve robot performance.

Several power sources tests are performed in order to study their effect on battery lifetime. The standard battery lifetime is shown by battery test. However, adding a supprcaacitor affects battery lifetime badly but it offers an ability to capture the flowed back energy from the motor. Moreover, the difference in battery lifetime from the standard case is not big. Replacing supprcaacitor by fuel cell shows good improvement in battery life time since the fuel cell supplies most of the load demand. However, this configuration is not able to recover motors energy. Finally, the best battery lifetime achieved is seen in case of having a battery connected in parallel conflagration to a fuel cell and supercapacitor, where the fuel cell still supplies the higher amount of current to the load and the supercapacitor gives to the ability to recover any energy produced by the motors. The cost for such enhancement can be seen in Fig. 106, where the fuel cell, GPS sensor, and robot platform are the main costly parts. Further, the cost breakdown can be found in Table 15.

As a future work, a DC/DC converter can be placed to control the amount of the energy supplied to the load. In supercapacitor case, the DC/DC converter will let the suppercapacitor connected to the bus when the motor produces energy. Otherwise,

charging/discharging current will be reduced by DC/DC converter since the supercapacitor draws current from other sources to charge itself. Usually, DC/DC converter is controller either by simple controller like PI or advanced controller, where the advance controllers are model dependent (e.g. GPC).

References

[1] B. V. Ratnakumar *et al.*, "Litmum batteries on 2003 mars exploration rover," in *Seventeenth Annual Battery Conference on Applications and Advances. Proceedings of Conference (Cat. No.02TH8576)*, Jan 2002, pp. 47–51.

[2] A. N. Wilhelm, B. W. Surgenor, and J. G. Pharoah, "Design and evaluation of a micro-fuel-cell-based power system for a mobile robot," *IEEE/ASME Transactions on Mechatronics*, vol. 11, no. 4, pp. 471–476, Aug 2006.

[3] A. C. Baisden and A. Emadi, "Advisor-based model of a battery and an ultracapacitor energy source for hybrid electric vehicles," *IEEE Transactions on Vehicular Technology*, vol. 53, no. 1, pp. 199–205, Jan 2004.

[4] A. Khaligh and Z. Li, "Battery, ultracapacitor, fuel cell, and hybrid energy storage systems for electric, hybrid electric, fuel cell, and plug-in hybrid electric vehicles: State of the art," *IEEE Transactions on Vehicular Technology*, vol. 59, no. 6, pp. 2806–2814, July 2010.

[5] S. M. Lukic *et al.*, "Power management of an ultracapacitor/battery hybrid energy storage system in an hev," in *2006 IEEE Vehicle Power and Propulsion Conference*, Sept 2006, pp. 1–6.

[6] S. M. Lukic *et al.*, "Energy storage systems for automotive applications," *IEEE Transactions on Industrial Electronics*, vol. 55, no. 6, pp. 2258–2267, June 2008.

[7] J. Cao and A. Emadi, "A new battery/ultracapacitor hybrid energy storage system for electric, hybrid, and plug-in hybrid electric vehicles," *IEEE Transactions on Power Electronics*, vol. 27, no. 1, pp. 122–132, Jan 2012.

[8] H. Banvait, S. Anwar, and Y. Chen, "A rule-based energy management strategy for plug-in hybrid electric vehicle (phev)," pp. 3938–3943, June 2009.

[9] R. Carter, A. Cruden, and P. J. Hall, "Optimizing for efficiency or battery life in a battery/supercapacitor electric vehicle," *IEEE Transactions on Vehicular Technology*, vol. 61, no. 4, pp. 1526–1533, May 2012.

[10] A. A. Ferreira *et al.*, "Energy management fuzzy logic supervisory for electric vehicle power supplies system," *IEEE Transactions on Power Electronics*, vol. 23, no. 1, pp. 107–115, Jan 2008.

[11] C.-C. Lin *et al.*, "Power management strategy for a parallel hybrid electric truck," *IEEE Transactions on Control Systems Technology*, vol. 11, no. 6, pp. 839–849, Nov 2003.

[12] J. Shen and A. Khaligh, "A supervisory energy management control strategy in a battery/ultracapacitor hybrid energy storage system," *IEEE Transactions on Transportation Electrification*, vol. 1, no. 3, pp. 223–231, Oct 2015.

[13] H. A. Borhan *et al.*, "Predictive energy management of a power-split hybrid electric vehicle," pp. 3970–3976, June 2009.

[14] J. P. Torreglosa *et al.*, "Predictive control for the energy management of a fuel-cell–battery–supercapacitor tramway," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 1, pp. 276–285, Feb 2014.

[15] H. A. Borhan *et al.*, "Predictive energy management of a power-split hybrid electric vehicle," pp. 3970–3976, June 2009.

[16] J. Richalet *et al.*, "Model predictive heuristic control: Applications to industrial processes," *Automatica*, vol. 14, no. 5, pp. 413 – 428, 1978.

[17] C. Cutler and B. Ramaker, "Dynamic matrix control - a computer control algorithm," *Proceedings American Control Conference*, 1980.

[18] D. Clarke, C. Mohtadi, and P. Tuffs, "Generalized predictive control—part i. the basic algorithm," *Automatica*, vol. 23, no. 2, pp. 137 – 148, 1987.

[19] C. E. Garcia and A. Morshedi, "Quadratic programming solution of dynamic matrix control (qdmc)," *Chemical Engineering Communications*, vol. 46, no. 1-3, pp. 73–87, 1986.

[20] C. E. Garcia and M. Morari, "A unifying review and some new results," *Industrial and Engineering Chemistry Process Design and Development*, vol. 21, no. 2, pp. 308–323, 1982.

[21] H. Kim and K. G. Shin, "On dynamic reconfiguration of a large-scale battery system," in *2009 15th IEEE Real-Time and Embedded Technology and Applications Symposium*, April 2009, pp. 87–96.

[22] K. B. Hatzell, A. Sharma, and H. K. Fathy, "A survey of long-term health modeling, estimation, and control of lithium-ion batteries: Challenges and opportunities," in *2012 American Control Conference (ACC)*, June 2012, pp. 584–591.

[23] S. Piller, M. Perrin, and A. Jossen, "Methods for state-of-charge determination and their applications," *Journal of Power Sources*, vol. 96, no. 1, pp. 113 – 120, 2001, proceedings of the 22nd International Power Sources Symposium.

[24] J. Xiong *et al.*, "Failure detection for over-discharged li-ion batteries," in *2012 IEEE International Electric Vehicle Conference*, March 2012, pp. 1–5.

[25] A. Banaei and B. Fahimi, "Real time condition monitoring in li-ion batteries via battery impulse response," in *2010 IEEE Vehicle Power and Propulsion Conference*, Sept 2010, pp. 1–6.

[26] Y. G. Chao Wu, Chunbo Zhu and Y. Zhao, "A review on fault mechanism and diagnosis approach for li-ion batteries," in *Journal of Nanomaterials*, vol. 2015, 2015.

[27] G. L. Plett, "Extended kalman filtering for battery management systems of lipb-based hev battery packs: Part 1. background," *Journal of Power Sources*, vol. 134, no. 2, pp. 252 – 261, 2004.

[28] M. Chen and G. A. Rincon-Mora, "Accurate electrical battery model capable of predicting runtime and i-v performance," *IEEE Transactions on Energy Conversion*, vol. 21, no. 2, pp. 504–511, June 2006.

[29] D. Ali *et al.*, "Uas based li-ion battery model parameters estimation," *Control Engineering Practice*, vol. 66, pp. 126 – 145, 2017.

[30] H. M. Usman, S. Mukhopadhyay, and H. Rehman, "Universal adaptive stabilizer based optimization for li-ion battery model parameters estimation: An experimental study," *IEEE Access*, vol. 6, pp. 49 546–49 562, 2018.

[31] F. Zhang, Z. Shi, and S. Mukhopadhyay, "Robustness analysis for battery-supported cyber-physical systems," *ACM Trans. Embed. Comput. Syst.*, vol. 12, no. 3, pp. 69:1–69:27, Apr. 2013.

[32] S. Mukhopadhyay and F. Zhang, "A high-gain adaptive observer for detecting li-ion battery terminal voltage collapse," *Automatica*, vol. 50, no. 3, pp. 896 – 902, 2014.

[33] M. C. Knauff *et al.*, "Simulink model for hybrid power system test-bed," *IEEE Electric Ship Technologies Symposium*, pp. 421–427, May 2007.

[34] E. V. Solodovnik, "An analytic approach to the design of nonlinear time-varying control systems," *International Journal of Systems Science*, vol. 32, pp. 629–637, 10 2001.

[35] G. Collins, *Experiment 3, Characterization of DC Motor: Part 1*, Colorado State University, 2017.

[36] G. Collins, *Experiment 4, Characterization of DC Motor: Part 2*, Colorado State University, 2017.

[37] J. A. Rossiter, *Model-Based Predictive Control: A Practical Approach*. CRC Press, 2017.

[38] D. Clarke, C. Mohtadi, and P. Tuffs, "Generalized predictive control—part i. the basic algorithm," *Automatica*, vol. 23, no. 2, pp. 137 – 148, 1987.

[39] D. Clarke, C. Mohtadi, and P. Tuffs, "Generalized predictive control—part ii extensions and interpretations," *Automatica*, vol. 23, no. 2, pp. 149 – 160, 1987.

[40] D. Simon, *Optimal State Estimation: Kalman, H, and Nonlinear Approaches*. John Wiley Sons, 2006.

[41] M. L. V. F. Viktor Šlapák, Karol Kyslan and F. Durovský, "Finite control set model predictive speed control of a dc motor," *Hindawi Publishing Corporation Mathematical Problems in Engineering*, vol. 2016, pp. 1–10, 2016.

[42] B. d'Andréa Novel, G. Campion, and G. Bastin, "Control of nonholonomic wheeled mobile robots by state feedback linearization," *The International Journal of Robotics Research*, vol. 14, no. 6, pp. 543–559, 1995.

[43] *Datum Transformations of GPS Positions*, -blox ag, July 1999. [Online]. Available: https://microem.ru/files/2012/08/GPS.G1-X-00006.pdf

[44] Mathworks, "Fast fourier transform," https://www.mathworks.com/help/matlab/ref/fft.html#f83-998360_seealso, accessed: Mar 15, 2017.

[45] Mathworks, "Classify patterns with a shallow neural network," https://www.mathworks.com/help/deeplearning/gs/classify-patterns-with-a-neural-network.html, accessed: Oct 02, 2017.

[46] W. S. Mcculloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *Bulletin of Mathematical Biophysics*, vol. 5, pp. 115 – 133, 1943.

[47] M. T. Hagan *et al.*, *Neural Network Design*. PWS Publishing Co., 1996.

[48] S. Alawnah and A. Sagahyroon, "Modeling of smartphones' power using neural networks," *EURASIP Journal on Embedded Systems*, vol. 2017, 12 2017.

[49] S. M. D. A. Adil Khurram, Habibur Rehman, "Comparative analysis of integer-order and fractional-order proportional integral speed controllers for induction motor drive systems," *Journal of Power Electronics*, vol. 18, no. 3, pp. 723–735, 2018.

[50] Mathworks, "Convert state-space representation to transfer function," https://www.mathworks.com/help/control/ref/c2d.html, accessed: Sep 27, 2017.

[51] Mathworks, "Convert model from continuous to discrete time," https://www.mathworks.com/help/control/ref/c2d.html, accessed: Sep 27, 2017.

[52] Mathworks, "Matlab support package for arduino hardware," https://www.mathworks.com/help/supportpkg/arduinoio/index.html, accessed: Mar 07, 2017.

[53] M. A. Zagrobelny and J. B. Rawlings, "Identifying the uncertainty structure using maximum likelihood estimation," in *2015 American Control Conference (ACC)*, July 2015, pp. 422–427.

[54] D. A. Keim *et al.*, "Guest editorial: Special section on visual analytics," *IEEE Transactions on Visualization and Computer Graphics*, vol. 12, no. 6, pp. 1361–1362, Nov 2006.

[55] S. Peng *et al.*, "State of charge estimation of battery energy storage systems based on adaptive unscented kalman filter with a noise statistics estimator," *IEEE Access*, vol. 5, pp. 13 202–13 212, 2017.

[56] A. Huang and J. Friend, "Prevent lithium dendrite formation in rechargeable batteries through surface acoustic waves," in *2017 IEEE International Ultrasonics Symposium (IUS)*, Sept 2017, pp. 1–1.

Appendix

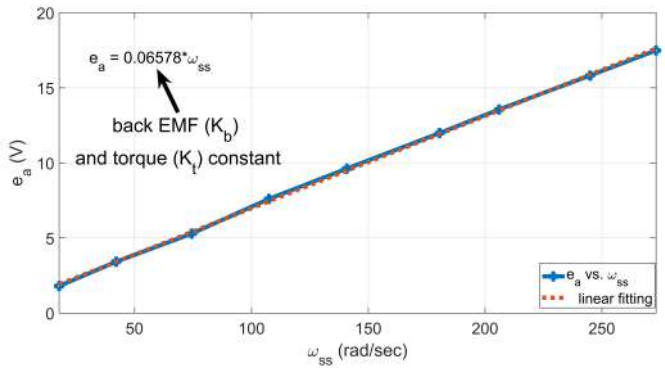

Figure 99: Back EMF (e_a) vs. steady-state angular speed (ω_{ss})

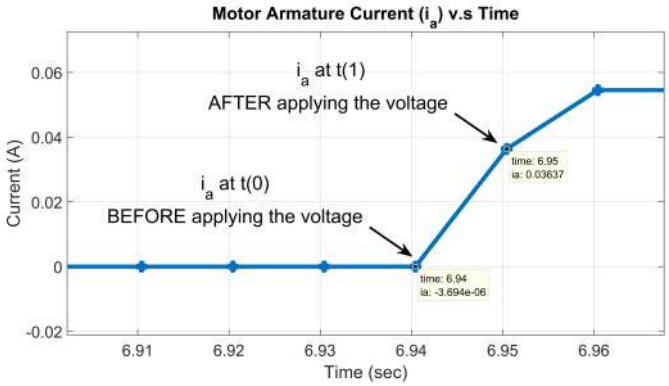

Figure 100: Motor armature current at time equal to zero

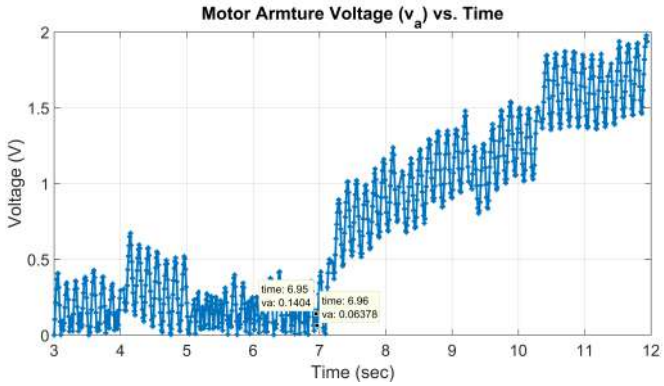

Figure 101: Motor armature voltage at time equal to zero

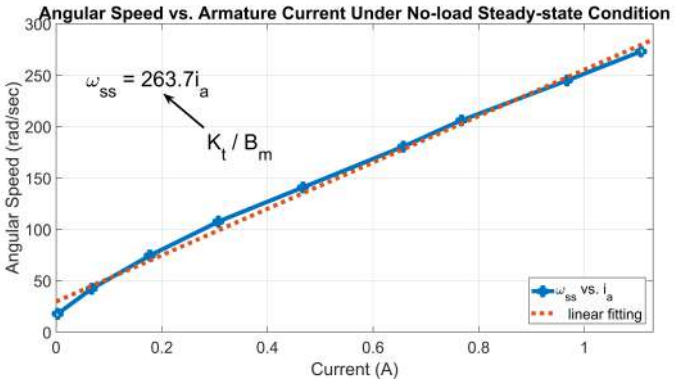

Figure 102: Speed vs. Current under no-load condition

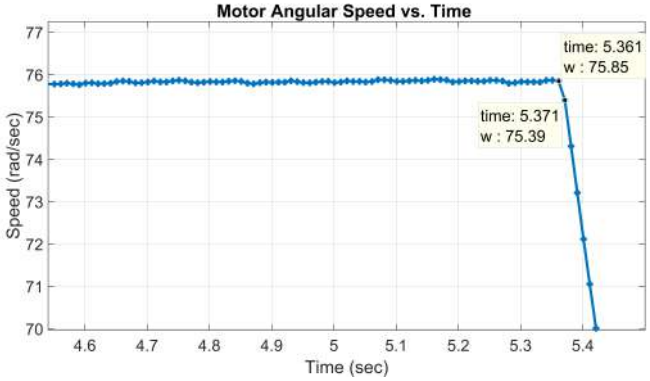

Figure 103: Shut downing the motor

Table 11: PMDC motor v_a, ω, and i_a under no-load steady-state condition.

Duty Cycle	v_a (V)	e_a (V)	ω_{ss} (rad/sec)	i_a (A)
0.55	2.4	1.8	17.7	0.0028
0.6	4.8	3.42	42.1	0.06
0.65	7.2	5.3	74.5	0.17
0.7	9.6	7.6	107.5	0.30
0.75	12	9.62	140.8	0.46
0.8	14.4	12	180.6	0.65
0.85	16.8	13.56	206	0.76
0.9	19.2	15.82	244.9	0.96
0.95	21.6	17.5	273.2	1.10

Table 12: PMDC motor v_a, ω, and i_a under locked-rotor condition.

Duty Cycle	v_a (V)	e_a (V)	ω_{ss} (rad/sec)	i_a (A)
0.55	2.4	1.21	0	0.02
0.6	4.8	2.53	0	0.3
0.65	7.2	4.2	0	1.05
0.7	9.6	5.3	0	2.3

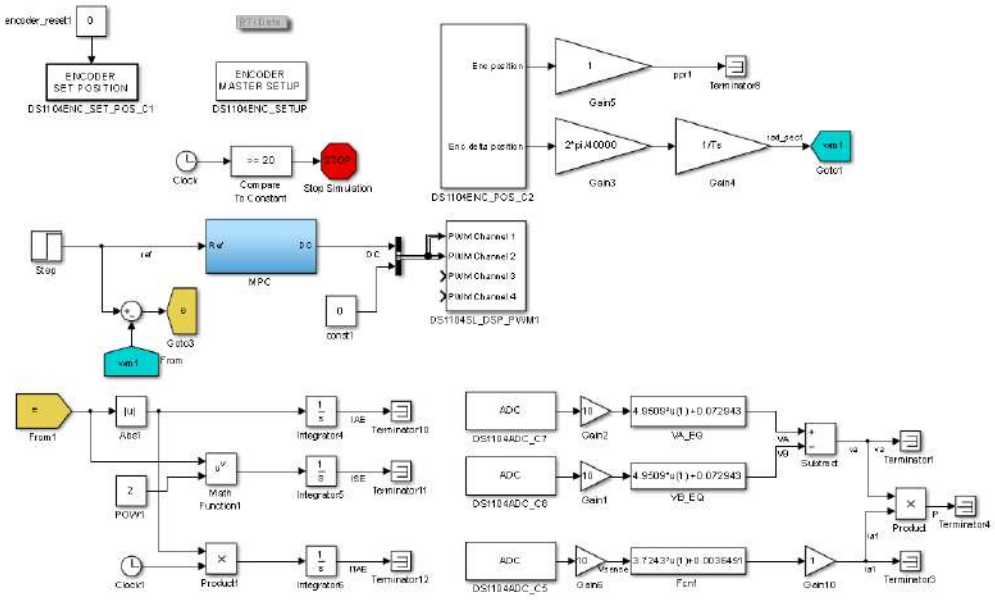

Figure 104: Full view GPC in MATLAB/Simulink environment

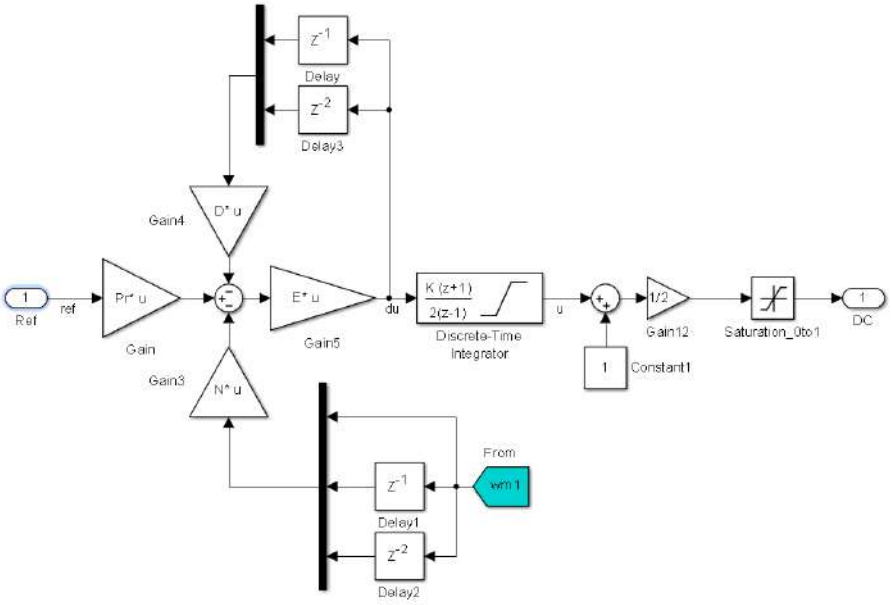

Figure 105: GPC in MATLAB/Simulink environment

Table 13: Robot PMDC motor parameters

Parameter	Symbol	Parameter Value	Unit
Resistance	R_a	1.13091922	Ohm
Back EMF constant	K_b	1.0146	Nm/rad/s
Torque constant	K_t	0.07358	Nm/A
Inductance	L_a	0.0345	H
Viscous friction	B_m	0.014	
Equivalent inertia	J_{eq}	0.0031	

Table 14: Comparison between different battery voltage collapse detection methods

Test No.	VT 22V	CT 10%	ANN $\gamma = 0.2$
Preliminary Test1	FA SOC=12%	TA SOC=10%	TA SOC=7.73%
Preliminary Test2	FA SOC=17%	TA SOC=10%	TA SOC=10%
Battery Test	FA Time=0.39 hr	LA Time=1.8 hr	TA Time=1.668 hr

Table 15: Cost for different robot items

Item	Qty.	Unit Price (AED)	Total Price (AED)	Remark
Robot platform	1	17,000	17,000	Robot platform includes chassis, 2 motors, 2 gearboxes, and 2 encoders.
PEM fuel cell + hydrogen cylinder	1	50,000	50,000	Manufacturer: Airostack
MIDG II INS/GPS sensor	1	25,000	25,000	
Labtop	1	2,300	2,300	
Li-ion battery	2	360	720	
Supercapacitor	1	180	180	Capacity: 3F
Current sensor	5	62	310	Commercial name: HXS 20-NP/SP30
Voltage sensor	4	12.5	50	
Motor driver	2	183	366	Commercial name: MegaMoto
Relay	4	10	40	SPDT type
Arduino board	2	360	720	Type: Mega2560
Total amount:			**96,686**	

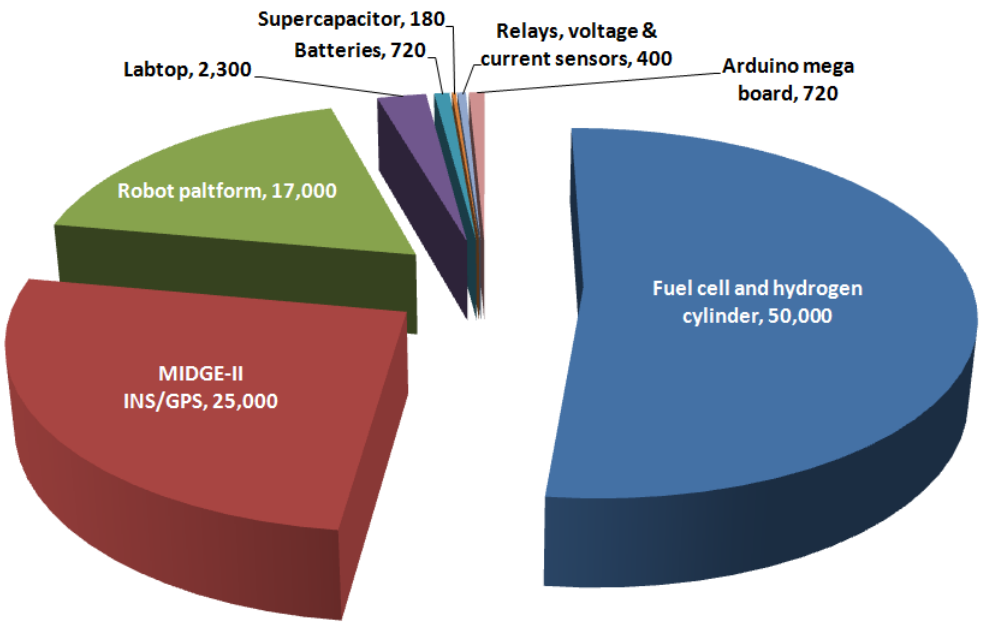

Figure 106: Robot cost in AED

Vita

Ali Qahtan Al-Tameemi was born in Basra, Iraq in 1990. He moved to United Arab Emirates in 2006, and finished his high school in 2008. Ali continued his education in Ajman University of Science and Technology and graduated with B.Sc in Electrical Engineering/Instrumentation and Control in 2014. Furthermore, he started his Mechatronics master program in American University of Sharjah in 2015.