EE-3001 Junior Design Studio
Project Report

Bike Overview:

The vehicle used for this project is Jolta Electric JE-70D. This project mainly evaluates the aforementioned bike's performance on different drive cycles. The electric bike is powered by a 20s-1p battery pack containing NMC cells. The pack is connected to BMS which constantly monitors cell-to-cell voltage and ensures safe and efficient operation of the bike. In addition, it protects the battery against overcharging and over-discharging. The positive terminal of the battery pack is connected to a circuit breaker, present to complete the circuit and effectively powers the bike when required. The inverter converts the DC power from this battery into AC power. Finally, the motor attached to the rear wheel takes the AC power from the inverter, converts the electrical energy to mechanical energy, and propels the bike forward. The control circuit is powered by a buck converter that constantly provides 12V DC converted into 5V using LM7805.

Project Summary:

This project aims at sensing the bike's battery terminal voltage, load current, and speed using Atmega 328 and storing it in an SD card for Data Logging. The current and voltage measurements are taken from the output of the battery terminal wires using the ACS712 current sensor and potential divider (of values 68kΩ and 4.7kΩ) respectively while its speed is taken from the hall effect sensor using the Arduino built-in pulseIn function. The results are then compared with actual measurements to determine accuracy. These measurements are then used to determine the performance of the bike on different drive cycles by driving the bike.

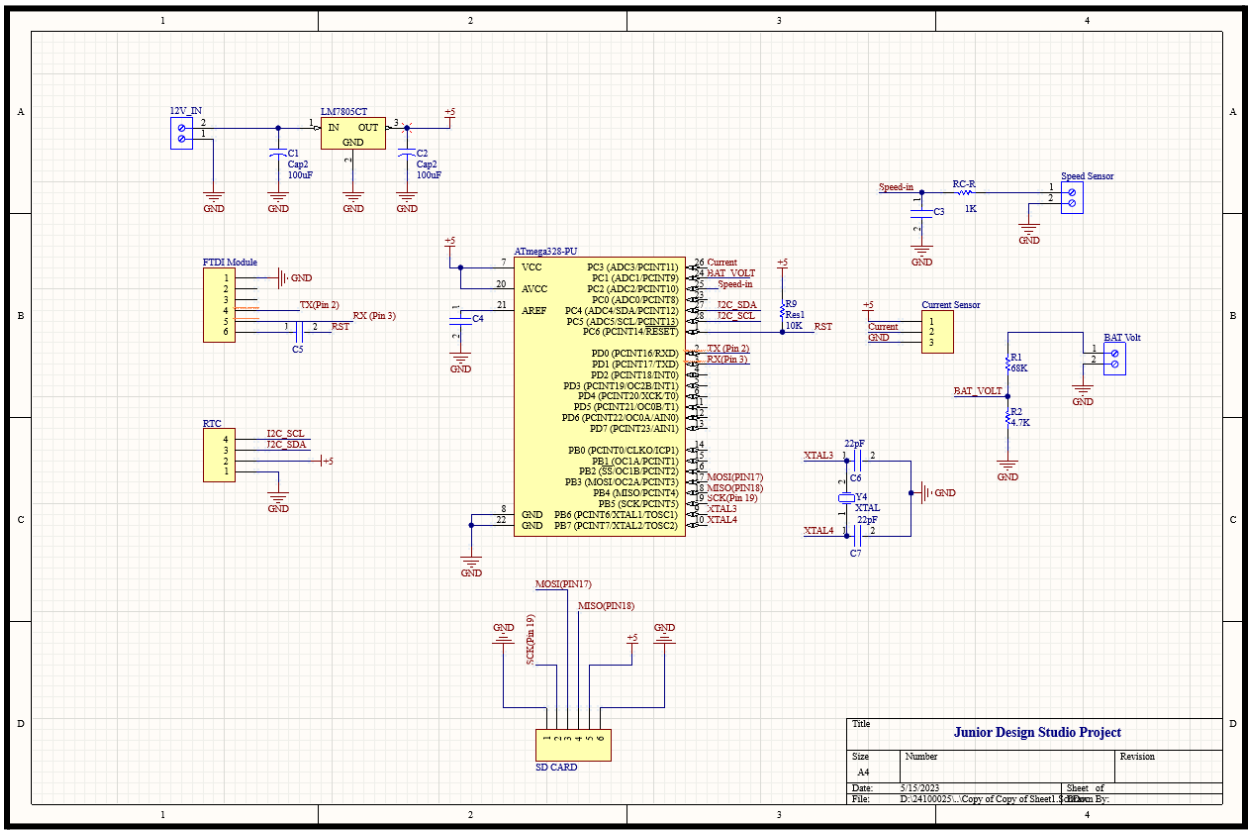
Parameter List:

- $\text{Capacity}_{\text{Cell}}$ = 17.82 Ah
- $\text{Weight}_{\text{Cell}}$ = 0.63 kg
- V_{nom} = 3.2 V
- # of $\text{Parallel}_{\text{Cell}}$ = 1
- # of $\text{Series}_{\text{Cell}}$ = 20
- $\text{Weight}_{\text{Pack}} = \text{Weight}_{\text{Module}}$ = 15.42 kg
- Pack SOC_{Max} = 80% (*Assumed*)
- Pack SOC_{Min} = 20% (*Assumed*)
- $\text{Efficiency}_{\text{Pack}}$ = 90% (*Assumed*)
- $\text{Torque}_{\text{Motor}}$ = *Can't be measured directly*
- $\text{Inertia}_{\text{Motor}}$ = *Can't be measured or calculated directly*
- $\text{Radius}_{\text{Wheel}}$ = 0.28 m
- $\text{Inertia}_{\text{Wheel}}$ = 0.392 kgm² ($\text{Radius}_{\text{Wheel}}^2 \times \text{Mass}_{\text{Wheel}}$)
- $\text{Roll Coefficient}_{\text{Wheel}}$ = *Depends on road conditions*

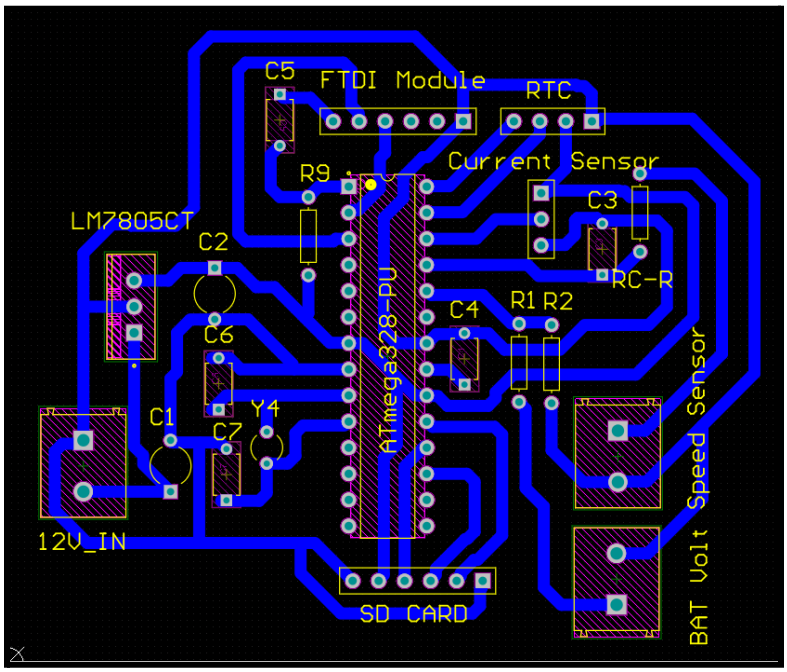
- $\text{Efficiency}_{\text{Inverter}} = \frac{P_{out}}{P_{in}} \times 100 = \frac{548.7\,W}{573.5\,W} \times 100 = 95.66\%$ (*Getting an unequal P_{out} across the 3 phases of the inverter. Manufacturing default*)
- Fractional Regen Torque = 0.93 (*From Variable File from Lab 2 Homework Assignment*)
- $\text{Ratio}_{\text{Gear}}$ = 1.0 (*Assumed)*
- $\text{Inertia}_{\text{Gear}}$ = 0.01 kgm^2 (*From Variable File from Lab 2 Homework Assignment*)
- $\text{Efficiency}_{\text{Gear}}$ = 98% (*From Variable File from Lab 2 Homework Assignment*)
- # of Wheels = 2
- $\text{Force}_{\text{Road}}$ = 0 N (*Assumed*)
- Drag Coefficient = 0.88 (*From Variable File from Lab 2 Homework Assignment*)
- Frontal Area = 1.292 m^2 (*Measured using built-in iPhone Measure App*)
- $\text{Weight}_{\text{Vehicle}}$ = 77 kg
- Payload = 75 kg
- $\text{Power}_{\text{Overhead}}$ = 56.151 W

Altium File Screenshots:

Schematic

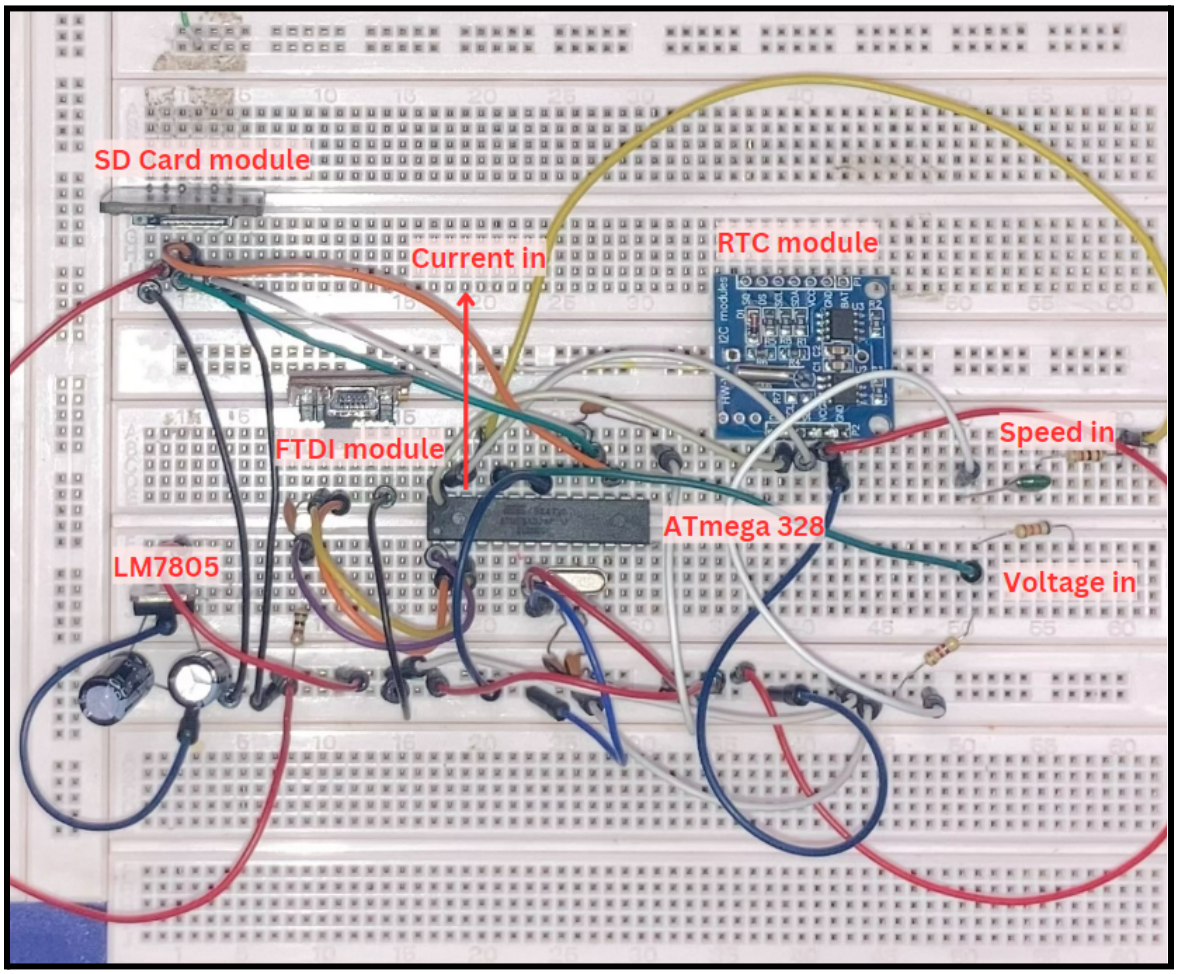


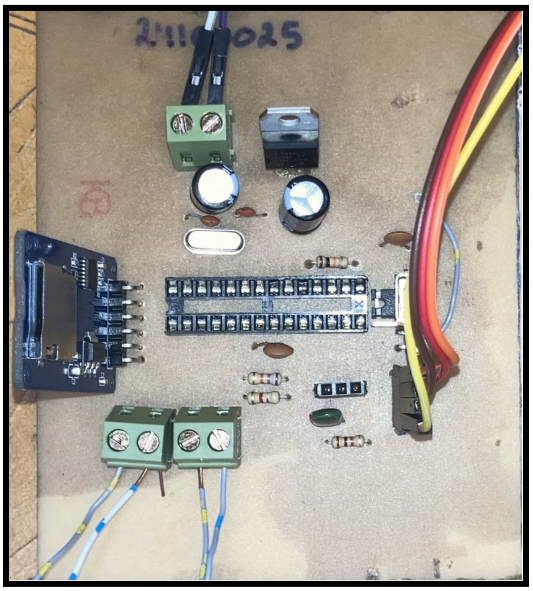
PCB Layout



Pictures of PCB / Breadboard:

Although we could not successfully implement our design on a PCB, we got reliable results using the breadboard implementation. The layouts are as follows:





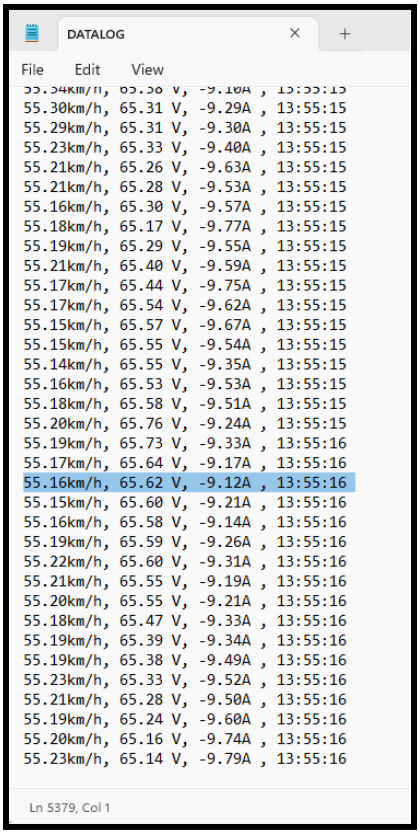
Complete Code:

```
#include "Arduino.h"
#include "uRTCLib.h"
#include "SPI.h"
#include "SD.h"

uRTCLib rtc(0x68);
const int chipSelect = 4;

const int hallSensorPin = A2;
const int voltageSensorPin = A1;
const int currentSensorPin = A3;

const float wheel_radius = 0.28;
const float current_senstivity = 0.175;
const int numReadings = 10;

float freqReadings[numReadings];
float voltageReadings[numReadings];
float currentReadings[numReadings];

float prevDuration1 = 0.0;
float prevDuration2 = 0.0;

float f_total = 0.0;
float v_total = 0.0;
float i_total = 0.0;

int readIndex = 0;

float get_speed();
float get_voltage();
float get_current();
String get_time();

void setup() {
  Serial.begin(9600);
  URTCLIB_WIRE.begin();
  pinMode(hallSensorPin, INPUT);
  while (!Serial);
  for (int thisReading = 0; thisReading < numReadings; thisReading++){
    freqReadings[thisReading] = 0.0;
    voltageReadings[thisReading] = 0.0;
    currentReadings[thisReading] = 0.0;
  }

  Serial.print("Initializing SD card...");
  if (!SD.begin(chipSelect)) {
    Serial.println("Card failed, or not present");
    while (1);
  }
  Serial.println("card initialized.");

  // For RTC Setup (Comment out once set)
  // rtc.set(second, minute, hour, dayOfWeek, dayOfMonth, month, year)
//  rtc.set(0, 46, 11, 2, 15, 5, 23);
}

void loop() {
  String dataString = "";
  if (readIndex >= numReadings){
    readIndex = 0;
  }
  float velocity = get_speed();
  float voltage = get_voltage();
  float current = get_current();
  String time = get_time();
  dataString += String(velocity) + "km/h, " + String(voltage) + " V, " + String(current) + "A , " +
time;
  File dataFile = SD.open("datalog.txt", FILE_WRITE);
  if (dataFile) {
    dataFile.println(dataString);
    dataFile.close();
```

```
    Serial.println(dataString);
  }
  else {
    Serial.println("error opening datalog.txt");
  }
  readIndex += 1;
}

float get_speed(){
  double pulseDuration1 = pulseIn(hallSensorPin, HIGH);
  double pulseDuration2 = pulseIn(hallSensorPin, LOW);
  if (pulseDuration1 != 0.0 && pulseDuration2 != 0.0) {
    if (pulseDuration1 < 50){
      pulseDuration1 = prevDuration1;
    }
    if (pulseDuration2 < 50){
      pulseDuration2 = prevDuration2;
    }
    prevDuration1 = pulseDuration1;
    prevDuration2 = pulseDuration2;
    double timePeriod = pulseDuration1 + pulseDuration2;
    double frequency = 1e6 / timePeriod;
    f_total -= freqReadings[readIndex];
    freqReadings[readIndex] = frequency;
    f_total += freqReadings[readIndex];
    float freq_avg = f_total / float(numReadings);
    float rpm = (2.339 * freq_avg) - 2.505;
    float vel = (rpm/60) * 2*PI * wheel_radius * 3.6;
    return vel;
  }
}

float get_voltage(){
  int value = analogRead(voltageSensorPin);
  double voltage = value * (63.9 /1023.0) * (72.7)/(68.0);
  v_total -= voltageReadings[readIndex];
  voltageReadings[readIndex] = voltage;
  v_total += voltageReadings[readIndex];
  float volt_avg = v_total/ float(numReadings);
  return volt_avg;
}

float get_current(){
  int value = analogRead(currentSensorPin);
  double current = ((value * (5.0 / 1023.0) - 2.96)/current_senstivity);
  i_total -= currentReadings[readIndex];
  currentReadings[readIndex] = current;
  i_total += currentReadings[readIndex];
  float current_avg = i_total/ float(numReadings);
  return current_avg;
}

String get_time(){
  rtc.refresh();
  String hour = String(rtc.hour());
  String minute = String(rtc.minute());
  String second = String(rtc.second());
  return hour + ":" + minute + ":" + second;
}
```

Data Log File:

The following values are taken directly from the DATALOG.txt file stored in SD card. The first column stores the linear velocity of the bike in km/h, while the second and third column display averaged-out battery terminal voltage and load current. The last column shows the timestamp of the uploaded values on the SD card.



Comparison with the measured values using certified instruments:

Note: We are comparing values of the highlighted portion of the DATALOG.txt file.

Conversion of measured rpm into linear velocity:

$RPM = 515\ rpm$

$Speed\ (km/h) = Radius_{Wheel} \times RPM \times \frac{2\pi}{60} \times \frac{3600}{1000} = 0.28 \times 515 \times \frac{2\pi}{60} \times \frac{3600}{1000}$

Speed (km/h) = 54.36

	Measurement	Datalog	% Error
Terminal Voltage (V)	63.5	65.62	3.34
Load Current (A)	-9.05	-9.12	0.773
Speed (km/h)	54.36	55.16	1.47

Actual Datalog File:

Note: The datalog file contains previous iterations of logging as well. The layout of the file is the same as explained above. The file is attached in the link below:

https://drive.google.com/drive/folders/1E8sZjlDQs502hCUmWiNisJkWbfl5Z4SG?usp=sharing