Universitá di Roma

TOR VERGATA

Master of Science in Mechatronics Engineering & Department of Electronic Engineering (2023-2024)

Project Name: Electronic IOT and Embedded Systems (EIES)

Mini-Elevator Project

Submitted by:

Earnest Ofonaike

Parth Bhalani

Artem Myshliaev

Submitted To:

Professor Giancarlo Orengo.

ii <u>Content</u>

iii. Forward

'*Any mechatronic student who cannot control a simple elevator system has got a problem on his hands*', Professor Riccardo Marino

The project became necessary when the statement was made during the course work of the control of mechanical system 2022/23. Therein the class, It resulted in assignment being given to student to model and simulate the elevator system.

The assignment became a challenge for modeling of this system. Base on the control analysis the system is of relative degree 2 and remain unstable such that there is a need to stabilize the plant, bring all the error to zero and bring the input to a constant value which is equivalent to the is disturbances (dead weight and friction) experienced.

When these problems were, to an extent, resolved theoretically, the doubt still exists that this simulation would not work as predicted with a physical model since there could be unforeseen variables. A group is students agreed to perform gap analysis by testing with physical model in the laboratory which necessitated initiation of this project – Mini Elevator Project.

1.0. Introduction

This project is the design and fabrication of a mini elevator system for the partial fulfilment of the course Electronic IOT and Embedded Systems (EIES). The word 'Mini' implies that the elevator system would be scaled to a portable size such that it can easily be moved into the electronic laboratory for all necessary test and evaluations, though scaled down dimensionally, it would beyond expectation, effectively give all the functionalities like that of a standards elevator.

Consequently, there a need to fabricate the mechanical parts which includes the carriage, the frame, and the pulley. Then install the Direct current (DC) motor, pullies, and the corresponding brackets. The softwarisation is implemented by the use of a microprocessor (Arduino Uno) and in addition to some other sensors to enable the required feedback and disturbance rejection.

The aim in brevity is to build the plant, implement control through micro-processor and smart sensors and then compare theoretical results with that of the physical results.

1.1. <u>The scope of Work</u>

- Features of the Elevator
- Dynamics and Control
- Design Realization
- Results and Inferences

2.0. Features and Design Realization

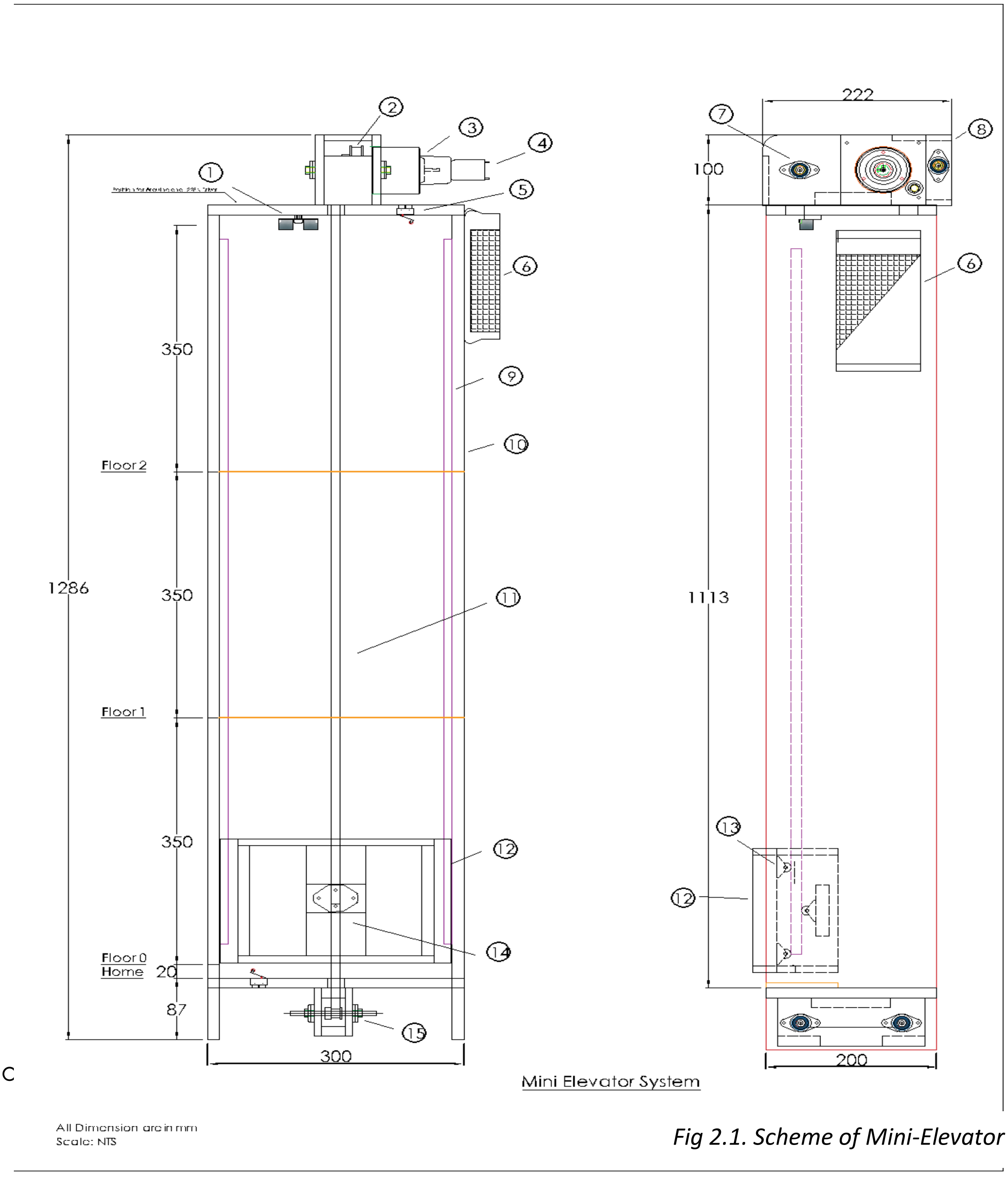
2.1. The features

The features of the sign can be categorized into three (3) parts which includes, Mechanical, Electronics analogue and Electronics digital systems. Each of these subsystems have components that obeys the physical laws of the designated to it performance properties. Below is a brief description of each category of feature and expected functionality.

2.2. Mechanical Components

The system dimension is bound in space by 3D dimension box of about [200, 300, 1300] millimeters. Containing a carriage restricted to move in all other direction but only in the vertical z-direction. The carriage is to be suspended by a cord able to bear the load while in motion or at its stationary state. The cord is connected to a speed reducer pulley driven by the DC motor also to be mounted at the top of the frame. The DC motor functions through a speed reducer pulley system and tout to avoid slip and backlash. There are also Idler pullies to ensure that the system runs smoothly.

And the frame, the stanchion and the brackets are made of wood. The structure is well braced to ensure stability.



The majc … below.

Fig 2.1. Scheme of Mini-Elevator

Component of the Mini Elevator		
Part No.	**Part Name**	**Description**
1	HC SR04 Ultrasonic Sensor	This is used to measure the distance covered by the system
2	Top Pulley System	This contains the sets of pinions and Idlers that pulls the belt and drive the carriage
3	30V DC motor	The is the system prime mover connect to 19mm pinion
4	Tachometer	A dynamo connects to the spin of the electric motor. Its output voltage magnitude can be interpreted as the speed of the motor
5	Limit Switches	Halts the movement of the carriage when it gets to permissible limits during movement
6	AC/DC Converter 240/24V	Supplied Power to the motor through the L298N Motor Driver
7	Pillow Bearing	Host the pin and hub of the idlers and pinions
8	Wooden bracket	Housing for the Pulley system and belts
9	Guide Rail	Guide the movement of the carriage along the z - direction
10	Stanchion	Body of the Elevators System. It is basically made of wood
11	Timing Belt _1	Drives the carriage and driven by 12mm diameter idlers. Timing Belt _2 drives the speed reducer system from the electric motor to the first Idler pulley.
12	Carriage	This is pulled by the belt and has the capability to carry loads
13	Roller For Carriage	Rolling wheels arranged to reduce friction between the carriage and the guide rail
14	Belt anchor & Lock	Anchors and locks the belt the belt to the carriage to enable the pulling
15	Base Pulley System	This contains the sets of Idlers that steadies the belt at the base as the carriage is driven

Table 2.1. Component List and Description of Mini-Elevator Scheme

2.3. Electrical Component

Under the electrical systems, there is a 30V DC rated electric motor and is powered by 24V DC supply from an AC/DC converter. We use the 24V DC power supply because the motor can operate between 10 to 30V and we do not require more that 12V power supply for all our operations. The elevator system does not operate at high speed, but at a low bandwidth such that gentle accelerator and deceleration is required for the comfort of the users.

The mechanical time constant of the motor is about 20 microseconds and we require far less than 200 microseconds. We do not expect any signal attenuation. (*Datasheet attached in the annex 1 for ready reference*)

The electric motor is fitted with a mini generator to serve as a tachometer which interprets the speed of the motor. The out voltage is measurable, through analogue input pin of the Arduino board.

Power is also supplied to the microprocessor - Arduino Uno by a dedicated 9-volt AC/DC converter and the latter in turn supplies the 5v to smart Ultrasonic distance sensor (HC SR04) while it is connected to the Arduino power output through a breadboard. The Arduino drives the motor by enabling the L298N DC motor Driver.

The Diagram below holds more information about the electrical connections.

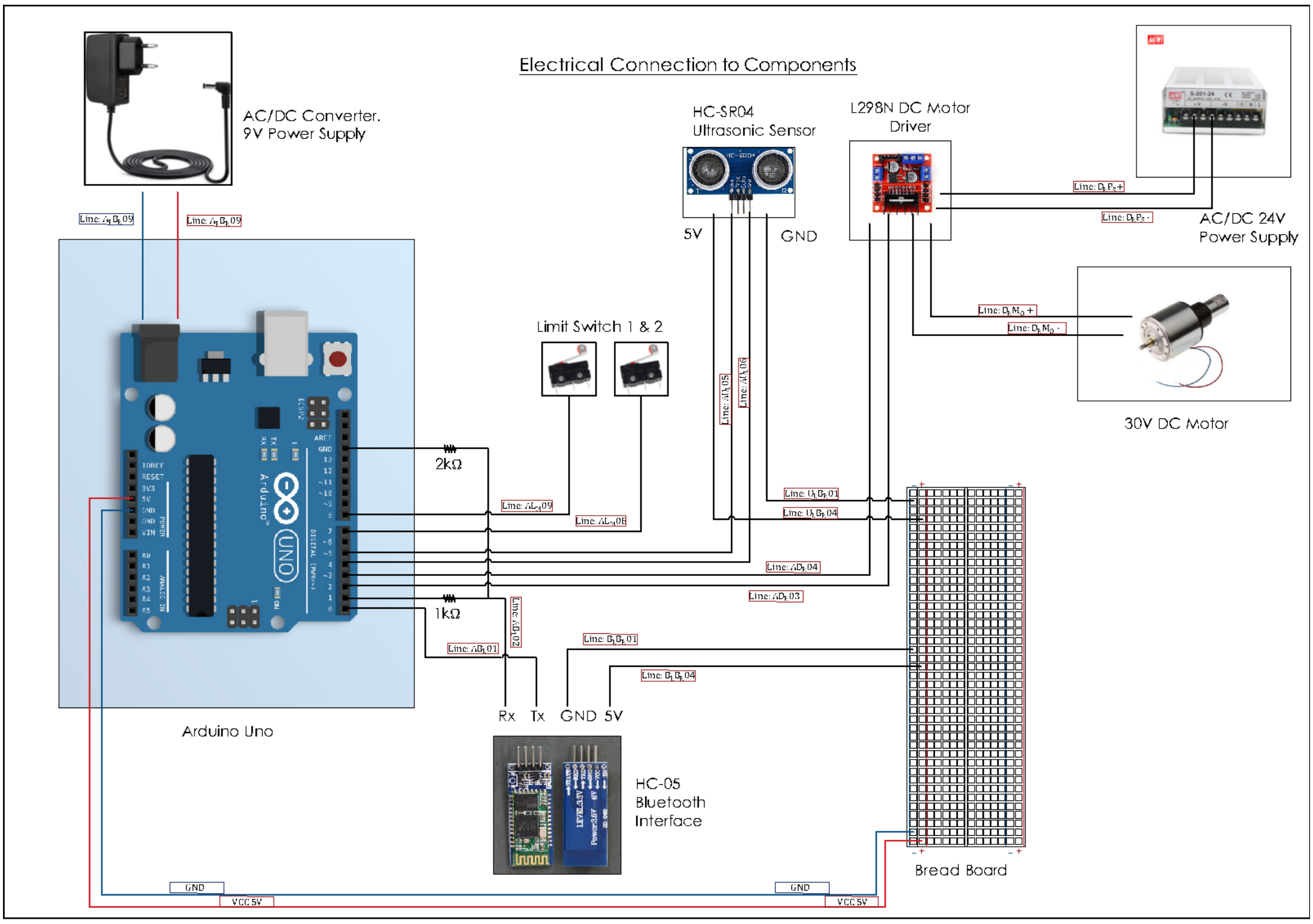


Fig 2.2. Electrical Circuits and Components

SR NO.	LINE CODES	DESCRPITIONS
1	ABL01	Line from Arduino (0) to RXD Bluetooth HC-05 (5V)
2	ABL02	Line from Arduino (1) to TXD Bluetooth HC-05 (3.3V)
3	BLBR01	Line from Bluetooth HC-05 (2) GND to GND Breadboard (0V)
4	BLBR04	Line from Bluetooth HC-05 (1) VCC to VCC Breadboard (5V)
5	ADR03	Line from Arduino (2) to IN2 Motor Driver (L298N)
6	ADR04	Line from Arduino (3) to IN1 Motor Driver (L298N)
7	DRMO	Line from Motor Driver L298N (1) to DC Motor
8	DRMO	Line from Motor Driver L298N (2) to DC Motor
9	DRPS V+	Line from Motor Driver L298N to V+ AC/DC Power supply(24V)
10	DRPS V-	Line from Motor Driver L298N to V- AC/DC Power supply(24V)
11	ALM09	Line from Arduino (8) to Limit switch 1
12	ALM08	Line from Arduino (7) to Limit switch 2
13	AUL05	Line from Arduino (4) to Echo (2) Ultrasonic sensor HC-SR04
14	AUL06	Line from Arduino (5) to Echo (3) Ultrasonic sensor HC-SR04
15	ULBR01	Line from Ultrasonic sensor HC-SR04 GND to GND Breadboard (0V)
16	ULBR04	Line from Ultrasonic sensor HC-SR04 VCC to VCC Breadboard (5V)
17	APBR09	Line from Arduino Analog pin (9) GND to GND Breadboard
18	APBR10	Line from Arduino Analog pin (10) GND to GND Breadboard

Table 2.2. Component List and Description for the Electrical Wiring

2.4. Electronics/ Digital Components

The microcontroller Arduino UNO is the central processing and controlling unit. It drives the system with L298N DC motor driver for speed and direction control. The feedback is the distance sensor wherein the HC-SR04 ultrasonic sensor is in place.

There is a HC - 05 Bluetooth asynchronous connection, from the central control to an android mobile device such that the control input is provided and monitored from the telephone.

3.0. Design Realization

3.1. The States

There are four states of operation of the system and in these states, there are specific operation which correspond to the operation of these states. The states are as follows: -

Off State: The system is powered down. The carriage is at the zero or ground floor, the off state can be initiated and the system powers off. If this state is imitated outside the ground floor, the carriage is moved to the ground floor before the shutdown.

On/Stationary state: This state is the stationary state. If the carriage is not moving, it is in this state. Once the system is switched on or get to desired floor, it goes to this state. In control this is the state wherein the control applies a holding torque, or the input becomes equivalent to the disturbances.

Ascent: The carriage travels in a direction against gravity in the positive z direction to a specified floor. The control is such that the system accelerates, travels at a given velocity and then decelerates gradually till it arrived at the desired floor. The control dynamics is divided into two based on the desired time for arrival to the different floors.

Descent: The descent is moving the same direction as gravity. In this case, the movement is not a free fall, controlled to move at the same pace like to the ascent.

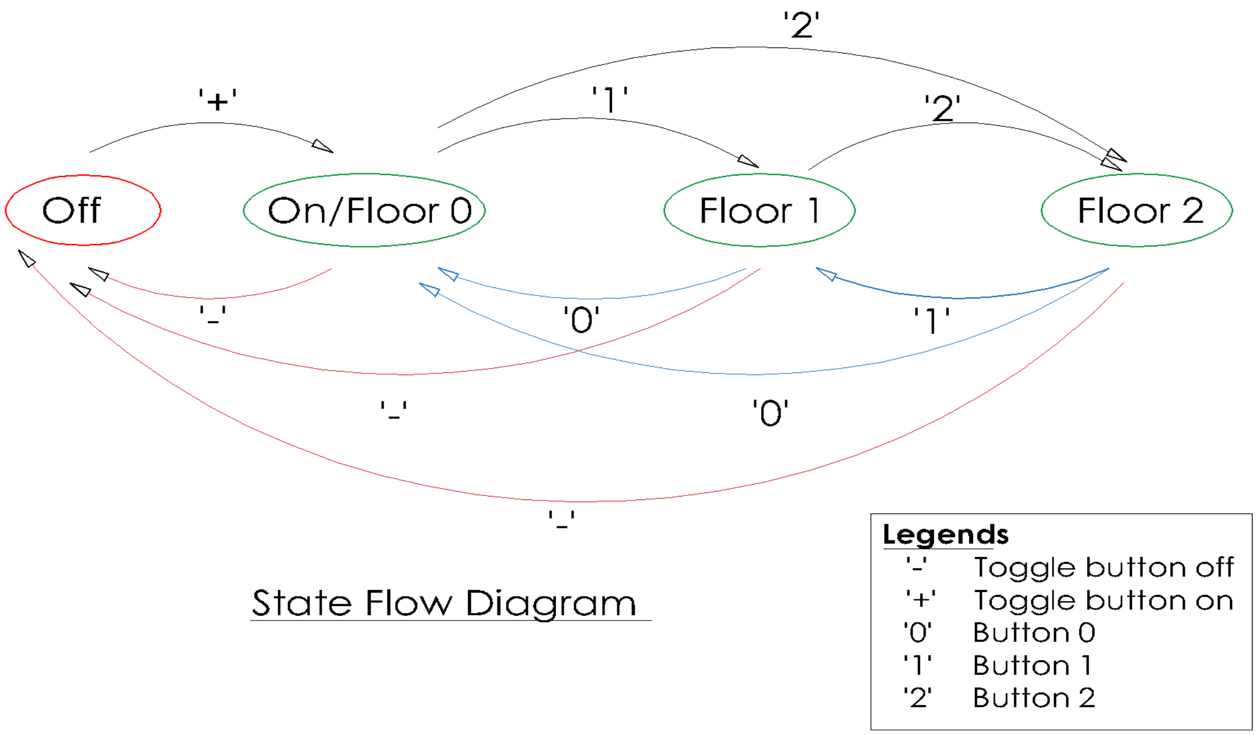


Fig 3.1. State flow schematic Diagram for Elevator Dynamics

3.2. Dynamics and control

The control was chosen such that the elevator gradually accelerates and decelerates to the comfort of the user. Considering the dynamic state, we have five (5) possible dynamics states which includes,

- Ascending single floor (+CI * 1)
- Ascending double floor (+CI * 2)
- Descending single floor (-CI * 1)
- Descending double floor (-CI * 2)
- Stationary position (balancing) – (Wt.)

Getting from to a subsequent from we design to take 6 second and two subsequent floor will take 8 seconds. And we allocate a 1 second max for estimation of disturbance (i.e. load). To enable the error to get to zero and the input to go to the value of the disturbance. We adopt the maximum velocity as 75mm/s.

3.3. Ascent

Ascending single floor (+CI * 1) and ascending double floor (+CI * 2)

$\underline{\boldsymbol{Dynamics}}$

$$y = k_0 + k_1 t + k_2 t^2 + k_3 t^3$$
$$\frac{dy}{dt} = v = k_1 + 2k_2 t + 3k_3 t^2$$
$$\frac{d^2y}{dt^2} = a = 2k_2 + 6k_3 t \qquad @\, k_1, k_2, k_3 = constants$$

$\underline{Boundary\ conditions}$

$$Acceleration\ a = 2k_2 + 6k_3 t$$
$$@t = 0, a = a_o\,;\ k_2 = \frac{a_o}{2}$$
$$@t = t_s, a = 0\,;\ k_3 = \frac{-a_o}{6t_s}$$
$$velocity\ v = k_1 + 2k_2 t + 3k_3 t^2$$
$$@t = 0, v = 0;\ k_1 = 0$$
$$v = a_o t - \frac{a_o}{2t_s} t^2$$
$$y = k_2 t^2 + k_3 t^3$$
$$@t = t_s, v = v_s;\ v = a_o t_s - \frac{a_o t_s}{2} = \frac{a_o}{2} t_s$$
$$\Rightarrow a_o = \frac{2v_s}{t_s} \quad - \qquad - \quad (i)$$
$$\boldsymbol{v = a_o t - \frac{a_o}{2t_s} t^2} \quad - \qquad - \quad (\boldsymbol{ii})$$
$$\boldsymbol{y = k_2 t^2 + k_3 t^3 = \frac{a_o}{2} t^2 - \frac{a_o}{6t_s} t^3} \qquad - \quad (\boldsymbol{iii})$$

$$Then\ during\ deceleration\ \ (k_2 = 0;\ k_1 = v_s)\ t_f = final\ time$$

$$y = k_0 + k_1 t + k_2 t^2 + k_3 t^3$$

$v = k_1 + 2k_2t + 3k_3t^2$

$v = v_s - \frac{a_o}{2t_f}t^2$

$y = v_st - \frac{a_o}{6t_f}t^3$

$@t = t_f, v = 0;\ a_o = \frac{2v_s}{t_f}$

$$\boldsymbol{v = v_s - \frac{a_o}{2t_f}t^2} \quad - \qquad\qquad - \quad (\boldsymbol{iv})$$

$$\boldsymbol{y = v_st - \frac{a_o}{6t_f}t^3} \quad - \qquad\qquad - \quad (\boldsymbol{v})$$

$\underline{Motion\ profile}$

$Given\ that\ the\ floor\ \boldsymbol{H = y_s + v_st_x + y_f}$

$\boldsymbol{y_s = f\ (t_s)\ \ and\ y_f = f\ (t_f)\ calling\ equation\ (iii)\ and\ (v)}$

$$\boldsymbol{H = \left(\frac{a_o}{2}t_s^2 - \frac{a_o}{6t_s}t_s^3\right) + v_st_x + \left(v_st_f - \frac{a_o}{6t_f}t_f^3\right)}$$

$Total\ time\ t = \boldsymbol{t_s + t_x + t_f\ = 6s\ \ also\ \ t_s = t_x = t_f\ = 2s}$

$$\boldsymbol{H = \left(\frac{a_o}{2}4 - \frac{a_o}{12} * 8\right) + 2v_s + \left(2v_s - \frac{a_o}{12} * 8\right)}$$

$$\boldsymbol{= \left(2a_o - \frac{2a_o}{3}\right) + 2v_s + \left(2v_s - \frac{2a_o}{3}\right)}$$

$\boldsymbol{calling\ equation\ \ (iv)}$

$\boldsymbol{v = v_s - \frac{a_o}{2t_f}t^2}; \quad @t = t_f, v = 0;\ a_o = \frac{2v_s}{t_f}$

$\therefore v_s = \frac{a_ot_f}{2} = \frac{2a_o}{2} = a_o$

$\Rightarrow \boldsymbol{H} = \left(\boldsymbol{2a_o - \frac{2a_o}{3}}\right) + 2a_o + \left(2a_o - \boldsymbol{\frac{2a_o}{3}}\right)$

$\boldsymbol{H} = \left(\boldsymbol{2a_o - \frac{2a_o}{3}}\right) + 2a_o + \left(2a_o - \boldsymbol{\frac{2a_o}{3}}\right)$

$\boldsymbol{H} = \boldsymbol{\frac{14a_o}{3}}\ @Height\ H = 350mm$

$\boldsymbol{a_o = \frac{3H}{14}} = \boldsymbol{\frac{3 * 350}{14}} = 75mm/s^2$

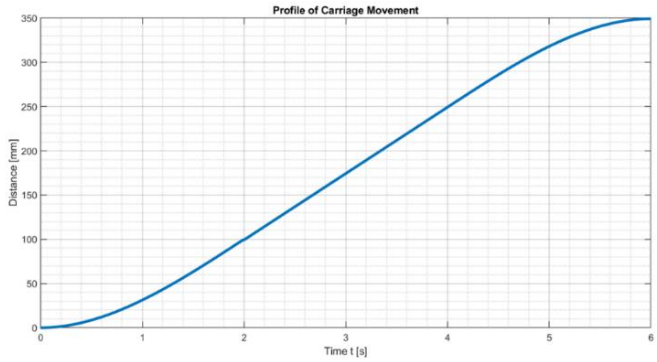


Fig 3.2. Distance – Time Profile in Millimetres

For computational convenience we give our distance in meters [m]

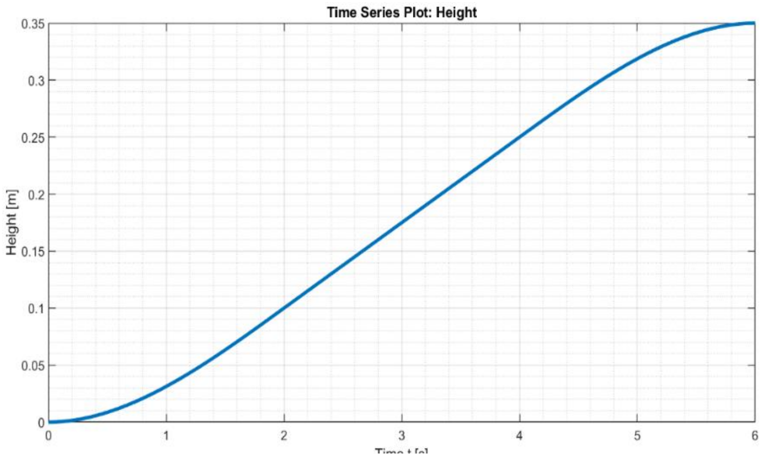


Fig 3.3. Distance – Time Profile in Metres

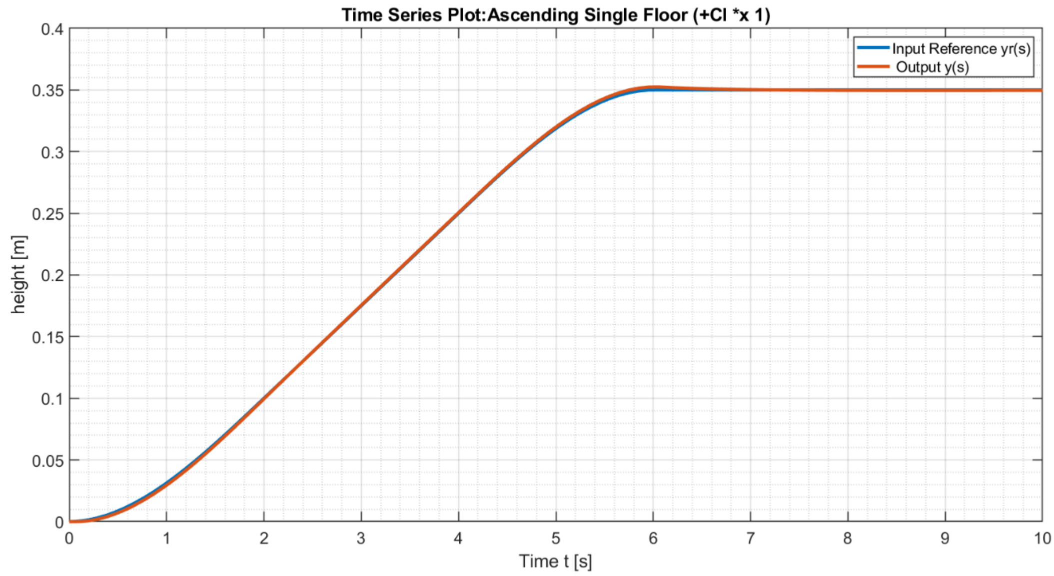


Fig 3.4. Design Performance profile – Height – Time Profile Ascent One floor

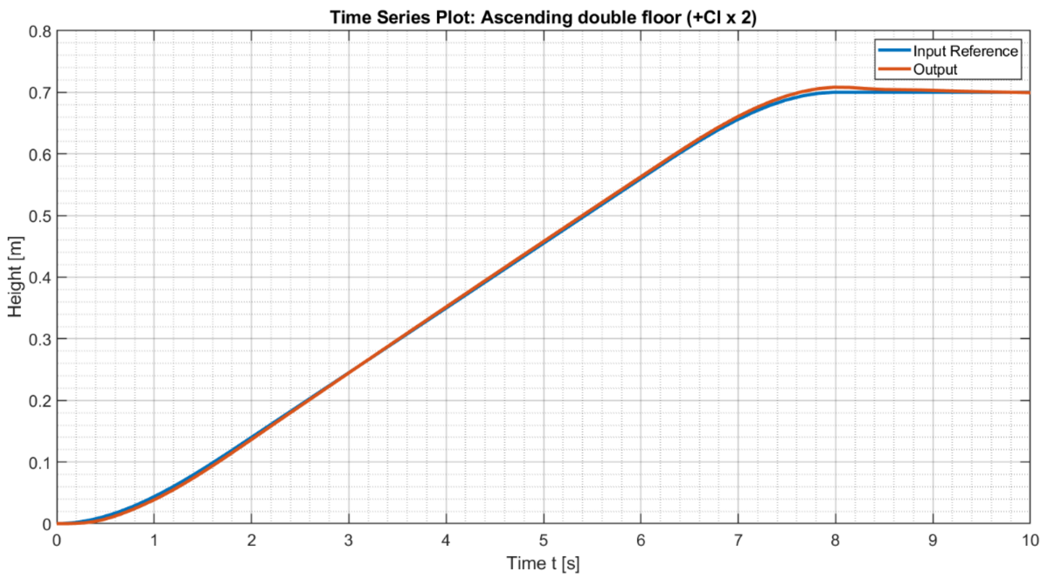


Fig 3.5. Design Performance profile – Height – Time Profile Ascent two floors.

3.4. Descent

Descending single floor (-CI * 1) and descending double floor (-CI * 2)

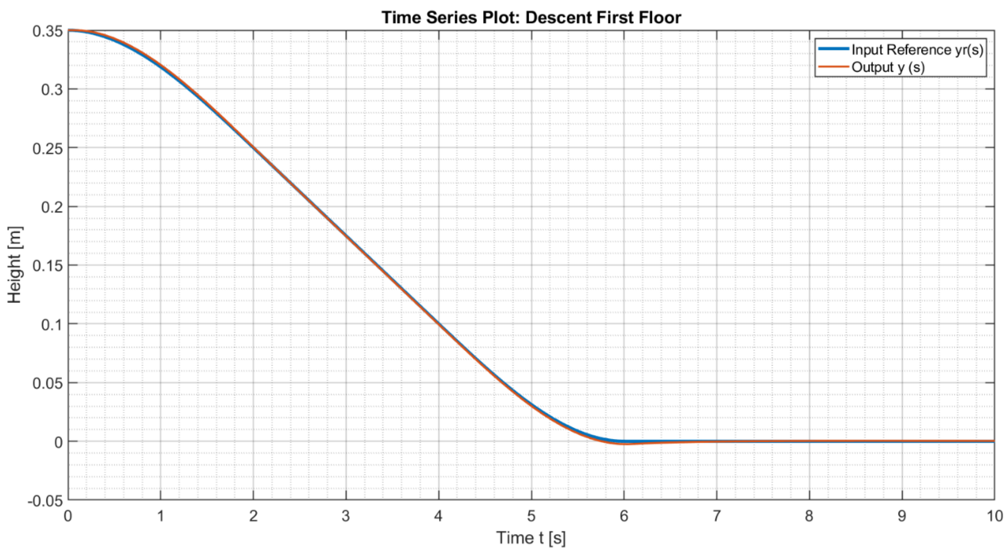


Fig 3.6. Design Performance profile – Height – Time Profile Descent One floor.

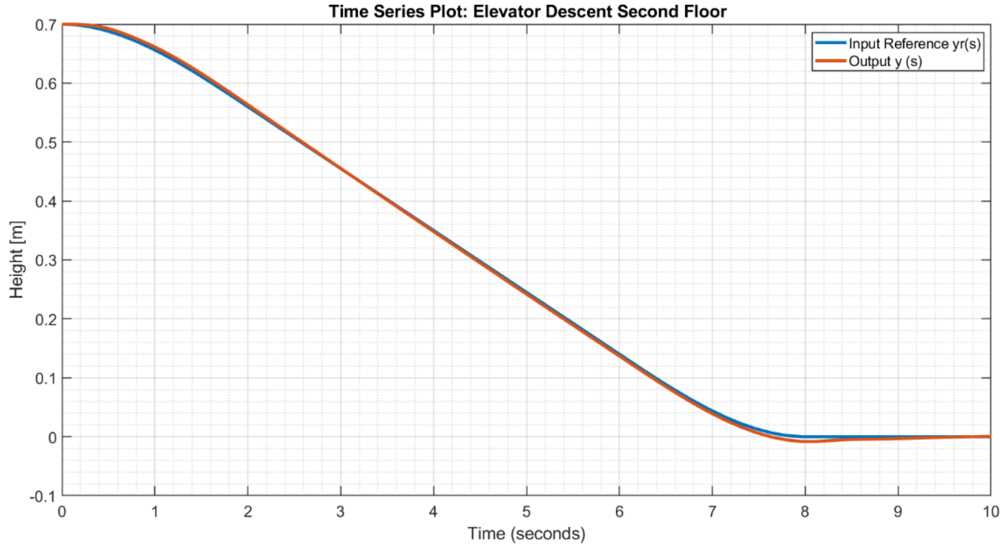


Fig 3.7. Design Performance profile: Height – Time Profile Descent Two floors

4.0. Algorithms and Programming

The main objective of the project - learn how to control a DC motor with Arduino sending commands through a Mobile App via Bluetooth. For solving this problem, the sketch for Arduino UNO board has been written and the mobile App has been developed in the MIT App Inventor.

- To simplify Speed Control of the motor, we decided to use the library "GyverMotor".
- Pulse Width Modulation approach was used to provide different speed.
- To avoid 'beeping' at low speed, we used 10-bit PWM: the voltage input is mapped from 0 to 1023;
- There are 3 simple functions to define basic motion of the motor: goUp(), goDown() and stop(). They transform a duty cycle of PWM signal (in %) to control input.
- To handle Limit Switches, we used another library "ezButton". The debounce time was added to prevent the button from being triggered again.
- Function 'getDist()' performs the distance measurement using Ultrasonic sensor. It sends the impulse to the carriage, then catches a reflected signal and calculates the time. Knowing the light speed, we can calculate the distance.
- The main loop consists of handling the commands from the app. The logic is quite simple. Since the movement down to Ground floor is the same in each case (stopping when the lower limit switch is pressed), we decided to isolate it into a separate function.
- If UP Limit Switch is touched the program stops.

4.1 The Arduino sketch:

```
#include <GyverMotor.h>                       // import the library to control the motor
(https://github.com/GyverLibs/GyverMotor)
#include <ezButton.h>                         // import the library to handle the limit switches presses
#define MOTOR_IN 2                            // define a digital pin №2 for the input motor signal
#define MOTOR_PWM 3                           // define a digital pin №3 for the input PWM motor signal
#define HC_ECHO 4                             // define a digital pin №4 for echo input signal from a Distance sensor HC-SR04
#define HC_TRIG 5                             // define a digital pin №5 for trigger output signal to a Distance sensor HC-SR04
#define LIM_SWITCH_DOWN 7                     // define a digital pin №7 for the input signal from the Bottom limit switch
#define LIM_SWITCH_UP 8                       // define a digital pin №8 for the input signal from the Upper limit switch
#define LED_mode 13                           // define a digital pin №13 for a built-in LED which indicates ON/OFF mode (ON
mode - LED ON, OFF mode - LED OFF)
#define LED_floor 12                          // define a digital pin №12 for a LED which indicates the floor when the mode is
ON (0 - LED ON, 1 - LED OFF, 2 - LED ON)

GMotor motor(DRIVER2WIRE, MOTOR_IN, MOTOR_PWM, HIGH);  // create 'GMotor' object which is controlled by a driver using 2 signals (pins:
MOTOR_IN, MOTOR_PWM)
ezButton limitSwitch_UP(LIM_SWITCH_UP);       // create 'ezButton' object for the Upper limit switch that attach to pin
LIM_SWITCH_UP;
ezButton limitSwitch_DOWN(LIM_SWITCH_DOWN);   // create 'ezButton' object for the Bottom limit switch that attach to pin
LIM_SWITCH_DOWN;

float const height0 = 82.0;                   // distance from the sensor to the GROUND floor
float const height1 = 51.5;                   // distance from the sensor to the FIRST floor
float const height2 = 18.0;                   // distance from the sensor to the SECOND floor

int state = -1;                               // elevator state: -1 (OFF mode - the elevator is anywhere), 0 (ON, Ground), 1
(ON, First), 2 (ON, Second)
char command = '-';                           // command from the app or a laptop: '-' (ON -> OFF), '+' (OFF -> ON, Ground), '0'
(1/2 -> 0), '1' (0/2 -> 1), '2' (0/1 -> 2)
```

```
float dist;                                         // distance to the elevator carriage that is calculated from the Distance sensor
measurements [function 'getDist()']
float err;                                          // distance error between the required floor and the current distance
bool flagON = false, flagOFF = false, flagRED = false; // flags that allow/forbid message output (to print the message only once)
uint32_t tmr_dist = 0;                              // time variable for storing the starting point.

void setup() {
  motor.setMode(AUTO);                              // set the motor operating mode (AUTO = automatic direction determination, others:
FORWARD, BACKWARD, STOP)

  motor.setResolution(10);                          // set 10-bit PWM mode: -1023..1023 (increasing the frequency helps to avoid
"beeping" at low speed)
  TCCR2B = 0b00000001;                              // set the Timer 1 (pins D3 & D11) to the 10-bit resolution
  TCCR2A = 0b00000001;

  Serial.begin(9600);                               // initialize the serial communication
  Serial.println("Start of the program!");

  pinMode(LED_mode, OUTPUT);                        // set the LED_mode pin as output
  pinMode(LED_floor, OUTPUT);                       // set the LED_floor pin as output
  pinMode(HC_TRIG, OUTPUT);                         // set the HC_TRIG pin as output
  pinMode(HC_ECHO, INPUT);                          // set the HC_ECHO pin as input

  limitSwitch_DOWN.setDebounceTime(50);             // set debounce time to 50 [ms] for each limit switch
  limitSwitch_UP.setDebounceTime(50);
}

/* -----------------------------------------------------------------------
ALGORITHM:
Inititally the system is in OFF mode (state = -1), the carriage can be anywhere.
OFF -> ON (command '+'): the elevator goes to the Ground floor (state = 0), stops and waits for a new command;
ON -> OFF (command '-'): the elevator stops (state = -1);
command '1': 0 -> 1 (the elevator goes up and stops at the 1 floor), 2 -> 1 (the elevator goes down and stops at the 1 floor);
command '2': 0/1 -> 2 (the elevator goes up and stops at the 2 floor);
```

```
command '0': 1/2 -> 0 (the elevator goes down and stops at the Ground floor);

To ensure precise positioning, the Feedback control was applied: the distance error drives to zero.
The downward movement continues until the bottom limit switch is pressed [function 'goHome()'].
RED BUTTON: to stop the system quickly, it is sufficient to press the Upper limit switch.
--------------------------------------------------------------------------- */

void loop() {
//Continuous data update
  limitSwitch_DOWN.loop();                             // continuously read the signals from the Limit switches
  limitSwitch_UP.loop();
  if (millis() - tmr_dist >= 100) {                    // read the distance from the sensor HC-SR04 every 100 [ms]
    tmr_dist = millis();                               // reset the starting point (function 'millis()' returns number of milliseconds
passed since the program started)
    dist = getDist();                                  // get the distance from the sensor
  }

//Reading commands from the Serial port or the RED BUTTON
  if (limitSwitch_UP.getState() == LOW) {              // RED BUTTON: if Upper limit switch is pressed, the system will switch to OFF
mode
    command = '-';                                     // this command will stop the motor
    if (flagRED == false) {                            // print (once!) that the Red Button was pressed
      Serial.println("RED BUTTON!");
      flagRED = true;
    }
  }
  else if (Serial.available()) {                       // read the command from the Serial port: Bluetooth (a Mobile app) or USB (a
laptop)
    command = Serial.read();
  }

//Command handling according to the elevator state
  if (command == '-') {                                // ON -> OFF
    stop();                                            // stop the motor
```

```
  digitalWrite(LED_mode, LOW);                         // turn off the LED to indicate that the elevator is NOT active
  digitalWrite(LED_floor, LOW);                        // turn off the LED which indicates the floor
  state = -1;                                          // switch to OFF mode
  if (flagOFF == false) {                              // print (once!) that the Mode was changed
    Serial.println("OFF mode");
    flagOFF = true;
    flagON = false;                                    // reset the flag to activate an ability to print messages about ON mode
  }
}

else if (command == '+' && state == -1) {              // OFF -> ON
  goHome();                                            // go down to the Ground floor (state = 0, command = '0')
  digitalWrite(LED_mode, HIGH);                        // turn on the LED to indicate that the elevator is active
  if (flagON == false) {                               // print (once!) that the Mode was changed
    Serial.println("ON mode");
    flagON = true;
    flagOFF = false;                                   // reset the flags to activate an ability to print messages about OFF mode
    flagRED = false;
  }
}

else if (command == '0' && (state == 1 || state == 2)) {   // 1 or 2 -> 0
  goHome();                                            // go down to the Ground floor (state = 0, command = '0')
}

else if (command == '1') {
  if (state == 0) {                                    // 0 -> 1
    err = dist - height1;                              // calculate the distance error
    Serial.println(err);                               // print a curent value of the error
    if (err > 0) {goUp(err);}                          // go UP until the error is zero
    else {
      stop();                                          // stop the motor
      state = 1;
      digitalWrite(LED_floor, LOW);                    // turn off the LED to indicate that the elevator is on the 1-st floor
```

```
      Serial.println("1-st floor");                         // print that the elevator has reached the 1-st floor
    }
  }
  else if (state == 2) {                                    // 2 -> 1
    err = height1 - dist;                                   // calculate the distance error
    Serial.println(err);                                    // print a curent value of the error
    if (err > 0) {goDown(err);}                             // go DOWN until the error is zero
    else {
      stop();                                               // stop the motor
      state = 1;
      digitalWrite(LED_floor, LOW);                         // turn off the LED to indicate that the elevator is on the 1-st floor
      Serial.println("1-st floor");                         // print that the elevator has reached the 1-st floor
    }
  }
}

else if (command == '2' && (state == 0 || state == 1)) {  // 0 or 1 -> 2
  err = dist - height2;                                     // calculate the distance error
  Serial.println(err);                                      // print a curent value of the error
  if (err > 0) {goUp(err);}                                 // go UP until the error is zero
  else {
    stop();                                                 // stop the motor
    state = 2;
    digitalWrite(LED_floor, HIGH);                          // turn on the LED to indicate that the elevator is on the 2-nd floor
    Serial.println("2-nd floor");                           // print that the elevator has reached the 2-nd floor
  }
}
}

void goHome() {
  if (limitSwitch_DOWN.getState() == HIGH) {              // while the elevator doesn't reach the Bottom limit switch
    err = height0 - dist;                                   // calculate the distance error
    Serial.println(err);                                    // print a curent value of the error
    goDown(err);                                            // go DOWN until the error is zero
```

```
  }
  else {
    stop();                                          // stop the motor
    state = 0;
    command = '0';
    digitalWrite(LED_floor, HIGH);                   // turn on the LED to indicate that the elevator is on the GROUND floor
    Serial.println("Ground floor");                  // print that the elevator has reached the GROUND floor
  }
}

void goUp(float err) {                               // function to force the elevator go UP (with decceleration at the end)
  if (err > 10) {motor.setSpeed(u(9));}
  else if (err > 5) {motor.setSpeed(u(8));}          // as the elevator approaches the desired floor, it reduces the speed
  else {motor.setSpeed(u(7));}
}

void goDown(float err) {                             // function to force the elevator go DOWN (with decceleration at the end)
  if (err > 10) {motor.setSpeed(-u(8));}
  else if (err > 5) {motor.setSpeed(-u(7));}         // as the elevator approaches the desired floor, it reduces the speed
  else {motor.setSpeed(-u(6));}
}

void stop() {                                        // function to stop the motor
  motor.setSpeed(0);
}

float u(int duty_cycle) {                            // function to get a motor PWM signal (0..1023) by a duty cycle (in %)
  return (duty_cycle/100) * 1024;
}

float getDist() {                                    // function to get the Distance from HC-SR04 ultrasonic sensor (returns the
distance to the Elevator carriage)
  digitalWrite(HC_TRIG, HIGH);                       // set trigger
  delayMicroseconds(10);                             // trigger impulse duration = 10 [mcs]
```

```
digitalWrite(HC_TRIG, LOW);                        // reset trigger
static uint32_t duration = pulseIn(HC_ECHO, HIGH);   // measure the echo duration as forward/back time [mcs]
return ((duration / 2) / 29.1);                      // calculate the distance [cm]      }
```

4.2 The Mobile App

Fig 4.1. Implementation of MIT App for Arduino Android Control Interface

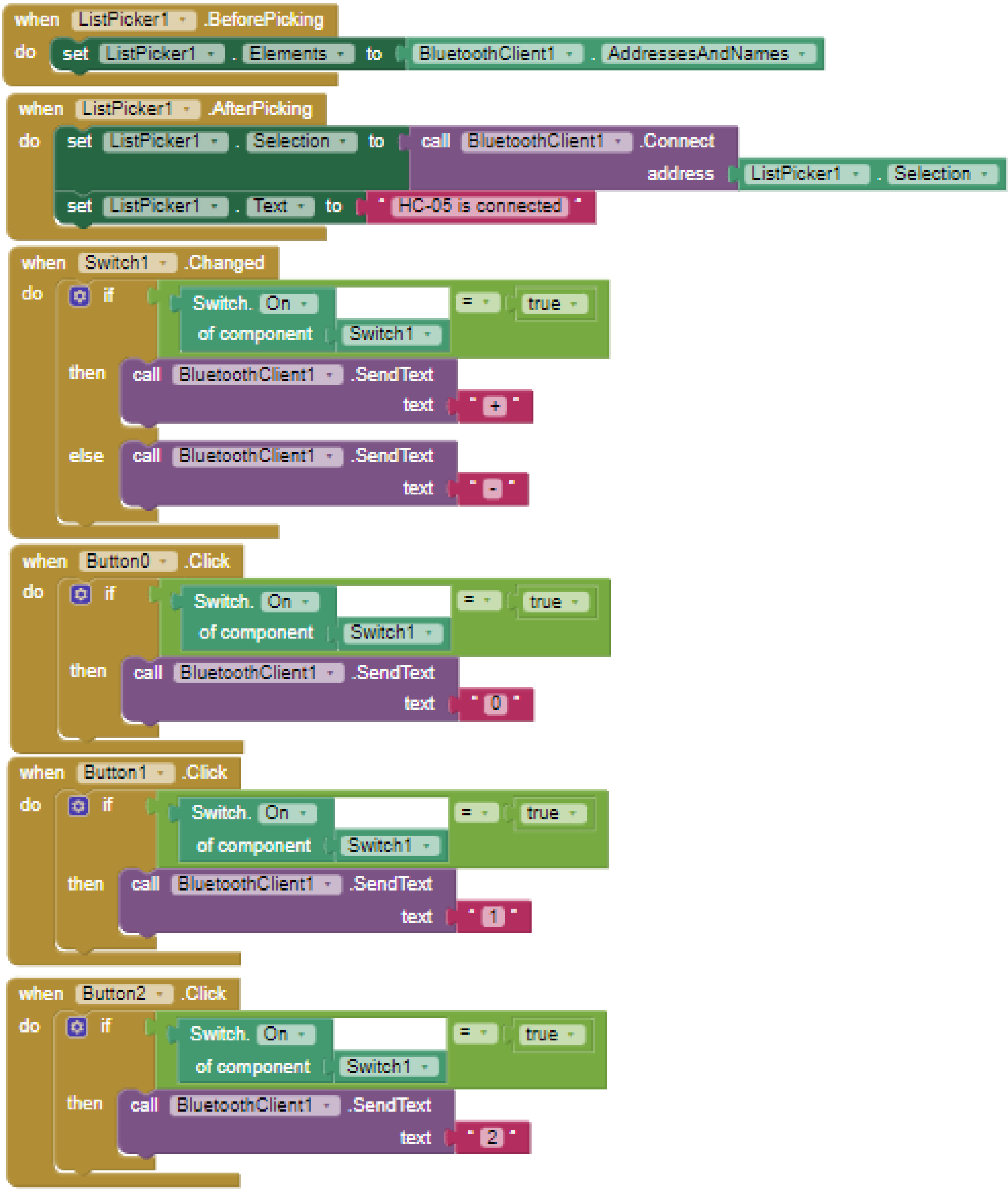


Fig 4.2. Implementation of MIT App for Arduino Android Control Interface - Expanded

The Mobile App comprises of 5 blocks: a ListPicker, a Switch and 3 buttons. List Picker aids in in connecting the smartphone to the Arduino Bluetooth Module choosing it from the list.

Changing the state of the Switch leads to sending the symbol '+' or '-'to Arduino Board. The floor buttons work only when the elevator is in ON mode.

5.0. Results and inferences

The elevator system performed as anticipated. It's acceleration and deceleration are gradual as per design and its carriages moves to the designated by inputting the reference floor on the android phone to the designated locations which represents the floor of interest.

We experience so system noise due to friction and some alignment error, but the system design with a feedback control system has an appreciable noise rejection capability. It simply brings the error signal to zero and also disturbances due to dead weight at the carriage and friction was also attenuated.

5.1. Conclusions

The possibility of system and sensor noise was witnessed and confirmed. The Arduino performed as predicted and to communicate wirelessly to remote device through a Bluetooth device.

The elevator, model can be recommended to be prototyped but with implementation of the fail-safe breaking system which we did achieved by using on a feedback design and hold torque from the DC electric motor.

The effect of friction and some parallax defects can be greatly reduced by fabricating components in a standard workshop as the elevator system was hand crafted just examine system noise rejection using feedback.

The elevator project has been completed successfully to the partial fulfilment of the course Internet of Things (IOT) and Embedded systems. The project reaffirms all that was theoretically instructed in the course work and performed as anticipated.

References

[1] Giancarlo Orengo. Lesson Notes "Prototyping Boards & Languages for IoT "Tor Vergata Roma A.A 2022/2033.

[2] O' Reilly and M. Margolis, "Arduino Cookbook" 2nd Edition; ISBN 978-1-449-31387-6, - December 2011

[3] RS PRO (St. No. 263 -6005)" Datasheet 30W servo Motor 24 – 30 V DC 14Ncm, 1600 rpm Motor," December 1999.

Annextures

Annex 1: Datasheet 30V Motor

ENGLISH

Datasheet

Stock No: 263-6005

RS Pro 30 W Servo Motor, 24 → 30 V dc, 14Ncm, 1600 rpm

Product Details

Servo Motor Fitted with Tacho

Fitted to servo motor (stock no. 263-5995) an integral dc tachogenerator to provide accurate velocity control, in applications which require smooth operation over a wide speed range

This motor is intended to be used with linear dc servo amplifier

Specifications

Maximum Output Torque	36 Ncm
Supply Voltage	24 → 30 V dc
Output Speed	1600 rpm
Power Rating	30 W
Stall Torque	14Ncm
Shaft Diameter	6mm
Length	134mm
Width	66mm

Instruction Leaflet
Bedienungsanleitung
Hojas de instrucciones
Foglio d'instruzioni

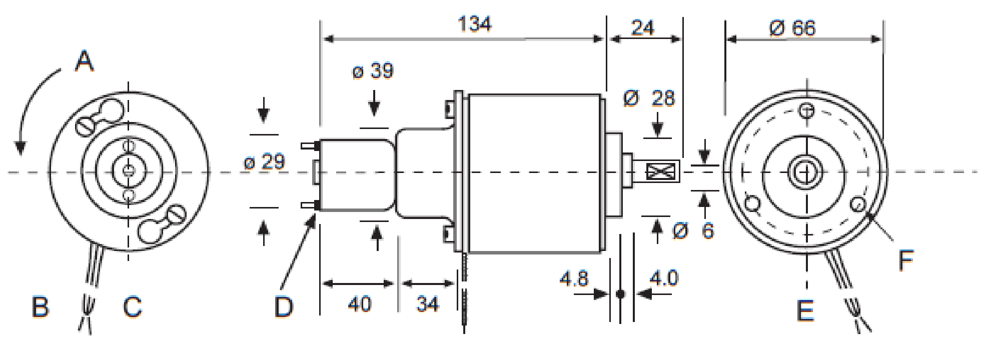
Low inertia dc servo motor-tacho unit (GB)

Trägheitsarmer DC-Servomotor mit Tachogenerator (D)

Unidad de servo dc de baja inercia y docificador (E)

Unità motore-tachimetro servo dc a inerzia ridotta (I)

Figures / Abbildung / Figura



GB

A. Direction of rotation with polarity shown
B. Red +
C. Blue -
D Tacho terminals
E. Leads 235mm typical
F. 3 fixing holes M4 x 8 deep equi-spaced on 50 mm PCD

Dimensions mm.

D

A. Drehrichtung mit Polarität
B. rot +
C. blau -
D. Tachoklemmen
E. Leiter normalerweise 235mm
F. 3 Befestigungslöcher M4 x 8 in jeweils 50mm Abstand

Abmessungen in mm

E

A. Dirección de rotación con polaridad indicada
B. Rojo +
C. Azul -
D. Bornes de tacómetro
E. Cables de conexión de 235mm normalmente
F. 3 orificios de fijación M4 x 8 de fondo separados igual distancia sobre PCB de 50mm

Dimensiones en mm.

I

A. Direzione di rotazione con polarità
B. Rosso +
C. Blu -
D. Terminali tachimetro
E. Terminali da 235mm (tip.)
F. 3 fori di fissaggio M4 x 8 a distanziati uniformemente su PCD da 50mm

Dimensioni mm

V10750

GB

RS Stock No.

263-6005

This unit utilises a high performance Low inertia dc servo motor with an integral dc tachogenerator to provide accurate velocity control in applications which require smooth operation over a wide speed range. Both the motor and tachogenerator incorporate skewed ironless rotors. The tachogenerator utilises precious metal commutation to provide a feedback signal with a high signal/noise ratio while the motor element utilises carbon brushes to provide long maintenance free life.

Technical specification

Maximum supply voltage	Vdc	40
Maximum continuous torque	Ncm	14
Maximum peak torque	Ncm	36
Motor voltage constant	V/1000 rpm.	10.3
Motor torque constant	Ncm/Amp	9.0
Mechanical time constant	ms	20
Rotor inertia	$kgcm^2$	0.214
Terminal resistance	Ohms	7.8
Rotor inductance	mH	5.0
Rotor construction		ironless
Commutation		carbon
Bearings		ball
Maximum axial force	N	15
Maximum radial force	N	100
Mass (motor-tacho assembly)	kg	1.03
Ambient temperature range		
Storage	deg.C	-40 to + 70
Operating	deg.C	-10 to + 60
Direction of rotation		reversible

Tachogenerator specification

Maximum ripple	Peak/peak	3%
Voltage tolerance		±10%
Ripple frequency	Cycles/rev.	18
Tacho assembly inertia	$kgcm^2$	0.011
Rotor construction		ironless
Commutation		Precious metal
Voltage constant	V/1000 rpm	3.25

Performance @ 24Vdc

No load speed	rpm	2300
Rated speed	rpm	1600
Rated torque	Ncm	12
Peak torque	Ncm	27

D

RS Best-Nr.

263-6005

Der hochleistungsfähige, trägheitsarme DC-Seromotor mit integriertem Gleichstrom-Tachogenerator für präzise Geschwindigkeitsregelung in Anwendungen, die ruhigen Betrieb über einen breiten Drehzahlbereich erfordern. Motor und Tachogenerator sind beide mit schrägen, eisenlosen Rotoren versehen. Der Edelmetall-Kommutator des Tachogenerators liefert ein Feedbacksignal mit einem hohen
Nutz-/Rauschsignalverhältnis, während der Motor mit seinen Kohlebürsten für lange, wartungsfreie Lebensdauer sorgt.

Technische Daten

Maximale Versorgungsspannung	V DC	40
Maximales Dauermoment	Ncm	14
Maximales Spitzenmoment	Ncm	36
Motorspannungskonstante	V/1000 min-1	10,3
Motordrehmomentkonstante	Ncm/A	9,0
Mechanische Zeitkonstante	ms	20
Rotorträgheit	$kgcm^2$	0,214
Klemmenwiderstand	Ohm	7,8
Rotorinduktanz	mH	5,0
Rotorausführung		eisenlos
Kommutator		Kohlebürste
Lagerausführung		Kugellager
Maximale Axialkraft	N	15
Maximale radiale Kraft	N	100
Gewicht (Motor mit Tachogenerator)	kg	1,03
Umgebungstemperatur		
Lager	°C	-40 bis + 70
Betrieb	°C	-10 bis + 60
Drehrichtung		umkehrbar

Technische Daten des Tachogenerators

Maximale Welligkeit	Spitze/Spitze	3%
Spannungstoleranz		±10%
Welligkeitsfrequenz	Zyklen/ Umdrehungen	18
Trägheit Tachoeinheit	$kgcm^2$	0,011
Rotorausführung		eisenlos
Kommutator		Edelmetall
Spannungskonstante	bei 1000 min-1	3,25

Leistung bei 24V DC

Leerlaufdrehzahl	min-1	2300
Nenndrehzahl	min-1	1600
Nennmoment	Ncm	12
Spitzenmoment	Ncm	27

E — Código RS.

263-6005

Esta Unidad emplea un servo motor dc de altas prestaciones y baja inercia con un codificador integrado de doble pista por incrementos, proporcionando un control digital preciso en aplicaciones que requieran un funcionamiento suave y control digital de la velocidad o la posición. El motor incluye un rotor desviado de material no ferroso que ofrece tanto un valor nulo de asincronía a baja velocidad como una rápida respuesta.
El codificador proporciona señales de salida TTL de doble pista de 500 pulsos por vuelta, lo que permite obtener una resolución en posición de hasta 2.000 cuentas por vuelta, siempre que se disponga de la electrónica posterior de medida adecuada.

Características técnicas

Tensión máx. de alimentación	Vdc	40
Par continuo máx.	Ncm	14
Pico máx. de par	Ncm	36
Constante de tensión de motor	V.1000 rpm.	10.3
Constante de par motor	Ncm/Amp	9.0
Constante mecánica de tiempo	ms	20
Inercia del rotor	$kgcm^2$	0.214
Resistencia de terminal	Ohms	7.8
Inductancia del rotor	mH	5.0
Material del rotor		no ferroso
Conmutación		carbono
Rodamientos		bolas
Fuerza axial máx.	N	15
Fuerza radial máx.	N	100
Masa (conj. motor-tacómetro)	kg	1.03
Margen de temperaturas		
Almacenamiento	°C	de -40 a + 70
Funcionamiento	°C	de -10 a + 60
Dirección de rotación		reversible

Características del tacogenerador

Fluctuación máxima	picos/picos	3%
Tolerancia de voltaje		10%
Frecuencia de fluctuación	ciclos /rev.	18
Inercia del conjunto del tacogenerador	$kgcm^2$	0,011
Construcción del rotor		sin armadura
Conmutación		metales preciosos
Constante de voltaje	V/1.000 rpm	3.25

Prestaciones a 24Vdc

Veloc. sin carga	rpm	2300
Veloc. nominal	rpm	1600
Par nominal	Ncm	12
Pico de par	Ncm	27

I — RS Codici.

263-6005

Questa Unità utilizza un motore servo dc a inerzia ridotta e ad alte prestazioni con un generatore-tachimetro dc integrale che garantisce una velocità elevata. È ideale per tutte le applicazioni nelle quali è necessario combinare fluidità di funzionamento e ampia gamma di velocità. Sia il motore che il generatore-tachimetro incorporano rotori ritorti senza parti in ferro.
Il generatore-tachimetro utilizza la commutazione in metallo prezioso per fornire un segnale di retroazione con un rapporto segnale/rumore particolarmente elevato. L'elemento motore, invece, utilizza spazzole al carbonio.

Specifiche tecniche

Tensione di alimentazione massima	Vdc	40
Coppia massima (continuo)	Ncm	14
Coppia massima (picco)	Ncm	36
Voltaggio costante motore	V/1000 giri/min.	10.3
Coppia motore (costante)	Ncm/Amp	9.0
Tempo meccanico (costante)	ms	20
Inerzia rotore	$kgcm^2$	0.214
Resistenza terminale	Ohm	7.8
Induttanza rotore	mH	5.0
Costruzione rotore		senza ferro
Commutazione		al carbonio
Cuscinetti		A sfere
Forza assiale massima	N	15
Forza radiale massima	N	100
Massa (motore-gruppo tachimetro)	kg	1.03
Gamma temperature ambiente		
Conservazione	deg.C	da -40 a + 70
Funzionamento	deg.C	da -10 a + 60
Direzione di rotazione		reversibile

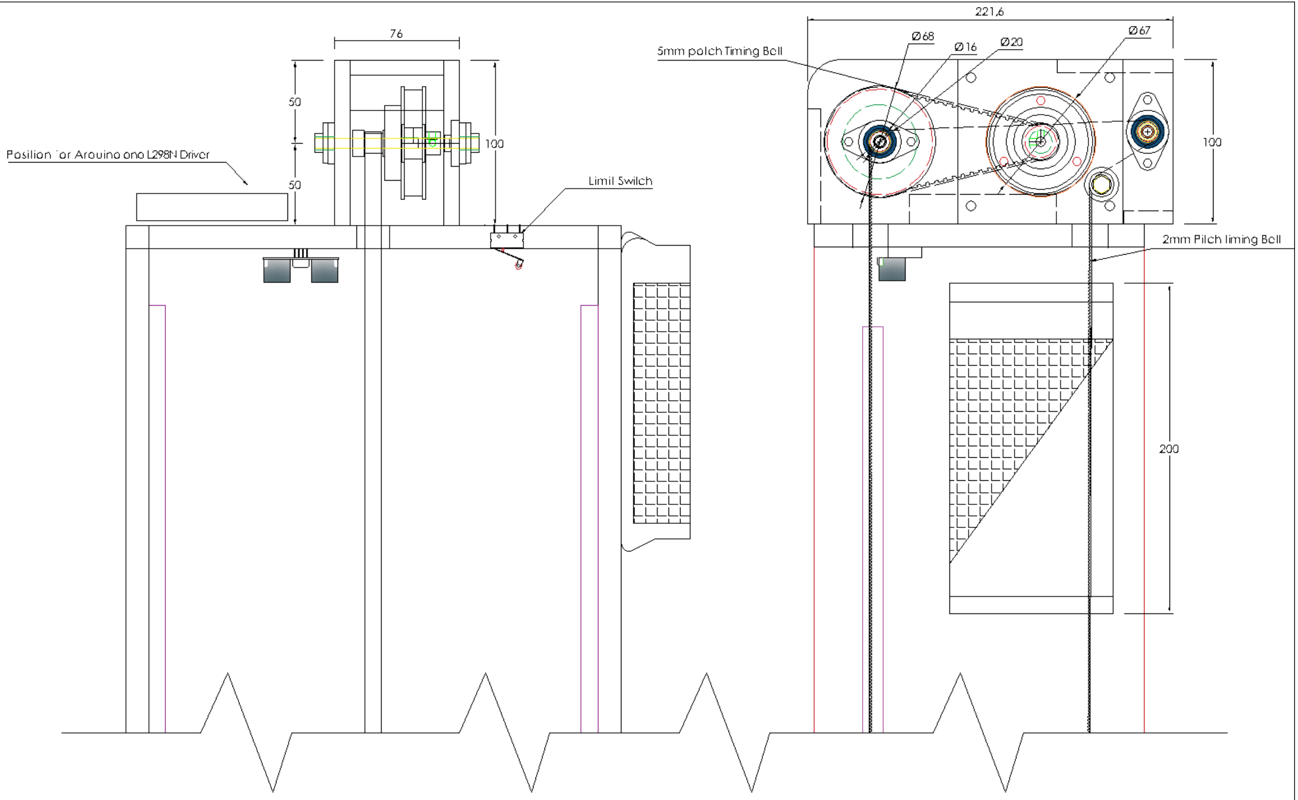
Specifiche Generatore Tachimetro

Ondulazione massima	Picco	3%
Tolleranza di tensione		±10%
Frequenza ondulazione	Cicli/riv.	18
Inerzia gruppo tachimetro	$kgcm^2$	0.011
Costruzione rotore		senza ferro
Commutazione		Metallo prezioso
Voltaggio (costante)	V/1000 giri/min.	3.25

Prestazioni @ 24Vdc

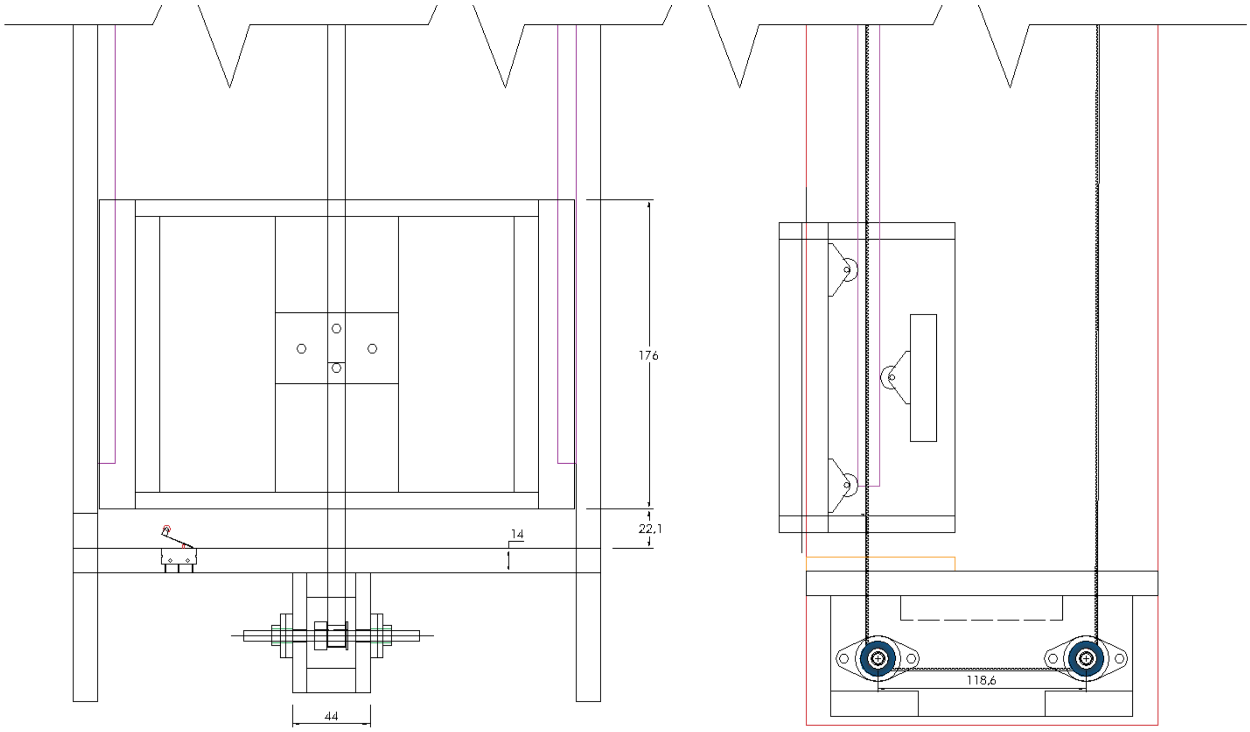
Nessuna velocità di carico	giri/min.	2300
Velocità nominale	giri/min.	1600
Coppia nominale	Ncm	12
Coppia (picco)	Ncm	27

Annex 2: Pulley Diagram of the Elevator



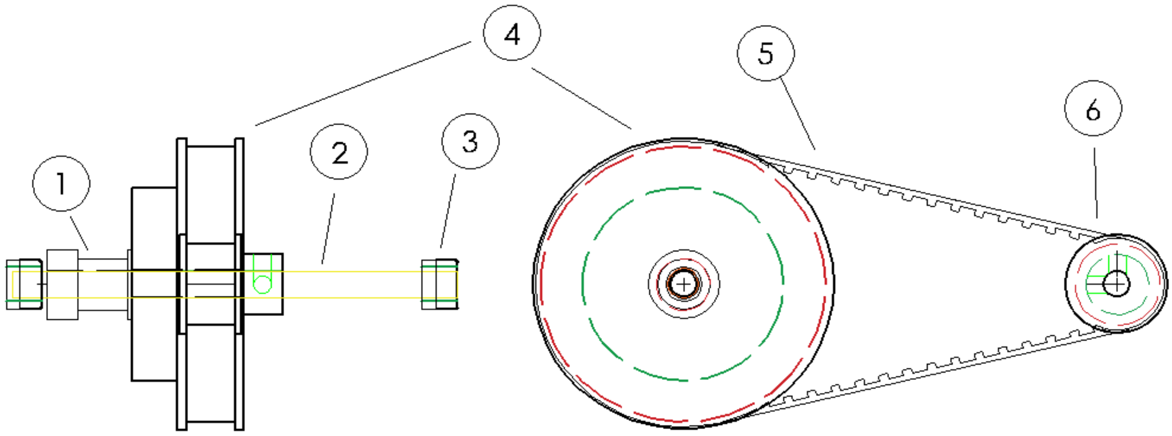
All Dimension are in mm
Scale: NTS

Top Positioned Component

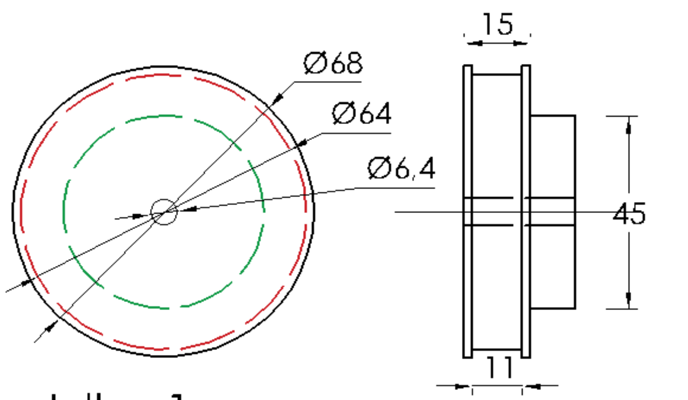


All Dimension are in mm
Scale: NTS

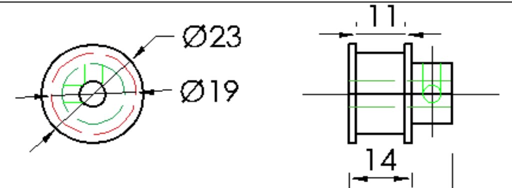
Base Side Component



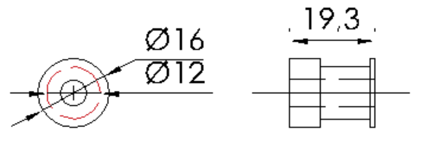
Pulley Components



Idler 1



Pinion



Idler 2

Mass Properties of the Pulley System				
S/N	Item	Volume [mm^3]	Moment of Inertia [mm^5]	Radius of Gyration [mm]
1	Idler 2	2,292.9	72,274.8	5.6
2	Pin	2,827.4	12,723.5	2.1
3	Collars	813.7	19,609.3	4.9
4	Idler 1	65,817.9	30,540,318.5	21.5
5	Belt	5,963.2		
	Pin+Collars+Idler(1&2)	**71,752.0**	**30,644,926.0**	**20.8**
6	**Pinion**	**4,046.8**	**254,150.9**	**7.1**

All Dimension are in mm
Scale: NTS