FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Tools and Control Experiments using TCLab Arduino Kit

Bruno Fernando Marques Aguiar

Master's in Electrical and Computer Engineering

Supervisor: Armando Jorge Miranda de Sousa

Supervisor: José Paulo Barroso de Moura Oliveira

July 30, 2020

Resumo

A educação no contexto de estudantes de nível universitário geralmente envolve muitas aulas teóricas de exposição longa, complementadas insuficientemente por uma componente menor de aprendizagem prática baseada em aulas experimentais. Atualmente, para melhorar a componente prática na área do controlo automático é necessária a utilização de equipamento laboratorial mais atual. O acesso a esse equipamento por parte dos alunos da Universidade nem sempre é possivel por razões várias. A menos que a carreira de um estudante o leve a conduzir pesquisas e não trabalhar numa empresa como a maioria, é provável que o referido trabalho tenha um forte foco pragmático e experimental. Isso mostra que, em geral, existe uma discrepância entre o que o mercado e a maioria dos empregadores procura em recém-formados e o que eles são capazes de fornecer,dadas algumas das limitações na sua educação. Para colmatar esta lacuna, existem alguns kits laboratoriais portáteis que permitem realizar experiências de controlo, com particular destaque para um kit baseado em Arduino: Temperature Control Laboratory (TCLab). Esses kits tentam resolver o problema da falta de experiência prática, fornecendo uma maneira de contornar os desafios tradicionais que as aulas de laboratório apresentam, os horários e os requisitos de equipamentos, permitindo conduzir experiências fora de prazos rígidos no conforto da casa de cada um. Além disso, as experiências que eles oferecem são pragmáticas e diretas ao assunto, pois são fáceis de usar e adequadas para iniciantes.

Neste documento, apresentam-se vários exemplos de dispositivos laboratoriais portáteis baseados no Arduino e Raspberry Pi que já foram testados, os quais são comparados em termos de acessibilidade, relevância didática e tipologias de abordagem adoptadas. As seguintes técnicas de controlo industrial foram implementadas e testadas no dispositivo TCLab: controlo on-off, controlo Proporcional, controlo Proporcional e Derivativo, Controlo Proporcional e Integrativo e Controlo Proporcional, Integrativo e Derivativo. Efetuou-se a identificação de um modelo de primeira ordem com atraso no tempo para o TCLab. Várias técnicas de sintonia de controladores PID baseadas nos parâmetros de um modelo de primeira ordem foram testadas no TCLab comparadas em termos de desempenho.

Abstract

Education in the context of university level students often involves a great deal of long theoretical exposition classes, complemented poorly by a smaller, or even non existent, component of practical learning based on experimental classes. Currently, to improve the practical component in the area of automatic control, the use of modern laboratory equipment is necessary. Access to this equipment by University students is not always possible for various reasons. Unless a students career path leads him/her to conduct research and not to work in a company like most do, it is likely that said work is going to have a strong pragmatic and experimental focus. This shows that, in general, there is a discrepancy between what the market and most employers are looking for in new college graduates and what they are able to provide given the limitations of their education. To fill this gap, there are some portable laboratory kits that allow control experiments to be carried out, with particular emphasis on an Arduino-based kit: Temperature Control Laboratory (TCLab). These kits try to solve the problem of a lack of "hands on" experience by providing a way to circumvent the traditional challenges that laboratory classes present ,schedules and equipment requirements, by allowing them to conduct experiments outside strict time frames in the comfort of their home. Besides, the experiments they offer are both pragmatic and straight to the point as they are easy to use and beginner friendly.

In this document various examples of take home laboratory kits are shown that were already tested to which it is compared regarding their accessibility, didactic relevance and approach typology. The following industrial control techniques were implemented and tested in the TCLab device: on-off control, Proportional control, Proportional and Derivative control, Proportional, and Integrative Control, Proportional, Integrative and Derivative control. A first order plus time delay model was identified for the TCLab. Several PID tuning techniques were tested in TClab and the performance obtained compared.

Acknowledgements

I would like to express my great appreciation to my supervisors, Paulo Oliveira and Armando Sousa, for their continued support and availability to help and advise me at any time I experienced any obstacles. For promoting a friendly, yet professional, work environment in which I could feel comfortable voicing my doubts and suggestions. Without them, the time spent making this dissertation wouldn't have been as constructive and pleasant as it was.
Additionally, I would like to thank my friends in university Mario Cardoso, Rafaela Oliveira and Francisco Neves in particular for having my back trough all these years and being a part of a cohesive team where everyone would try to contribute to the best of their ability when someone was struggling to achieve their goals.
I am deeply grateful to my parents for making this achievement possible in the first place, especially in times of uncertainty when they stood fast and didn't falter in their support and being understanding of my mistakes.
Finally, I cannot thank enough my girlfriend and colleague, Joana Cunha, for being by my side from the beginning and spending countless hours making sure we would both see the greatest challenge of our lives to it's end successfully. She made, what sometimes were difficult times, transform into the happiest so far just by being a kind, funny and lovable person.

Bruno Aguiar

"One must imagine Sisyphus happy."

Albert Camus

Contents

List of Figures

List of Tables

Acronyms and Symbols

TCLab	Temperature Control Laboratory
USB	Universal Serial Bus
PC	Personal Computer
PSO	Particle Swarm Optimisation
FOPTD	First Order Plus Time Delay
PCB	Printed Circuit Board
P	Proportional Controller
PI	Proportional Integrative Controller
PD	Proportional Derivative Controller
PID	Proportional Integrative and Derivative Controller
AMIGO	Approximate M-constrained Integral Gain Optimisation
SIMC	Simple Internal Model Control
IAE	Integral of the absolute value of the error
ITAE	Integral of the time-weighted absolute value of the error
TV	Total Variation
T1	Temperature reading in TCLab's sensor regarding heater 1
h1	Heater 1, TCLab actuator
SS	Steady-state
WWW	*World Wide Web*

Chapter 1

Introduction

1.1 Problem Specification

Introductory classes for control engineering require that students acquire both theoretical and practical concepts. Control engineering teaching requires a strong practical component with specific laboratory equipment however, this kind of hardware is quite expensive making it dependable on the quantity of students and number of feedback control classes in an University. Providing laboratory access to all courses is a difficult situation and when the institution lacks funds is even worse.Additionally, providing extra class access to those practical laboratories is a challenge. While the utilisation of simulations can help, it can never replace practical experiences with the methods discussed. One way around this issue is the use of remotely accessed laboratories. However, some of these experiments have some degree of complexity and are not very user-friendly for undergraduate students and therefore, easy to use and affordable control kits are highly desirable (both by lecturers and students) [1, 9].

Arduino [4] is wide spread also in the field of control teaching experiments. Unfortunately, many of the reported affordable applications require users to assemble their own experiment set and/or kit [10]. Thus, the combination of low cost solution/ready to be used and the availability of free supporting material is not easy to find.

Recently a very interesting kit fulfilling the former requirements was proposed by [7, 11, 12] for temperature control, named Temperature Control Laboratory (TCLab). Indeed, TCLab is a low-cost, ready to use portable take-home [1] which has been reported successfully in several teaching/learning experiments [13, 14, 15]. Thus, the TCLab is adopted in this work to conduct a series of control engineering related with the design of industrial controllers.

1.2 Objectives

It is well known that the following controllers are used within the clear majority of industrial control loops: On-Off and Proportional, Integrative and Derivative (PID). Thus, due to this practical relevance, it is mandatory its inclusion in introductory feedback control courses. Thus, practical laboratory experiments validating theoretical concepts regarding On-Off and PID control is quite relevant and are addressed in this work. The objectives of this thesis can be summarised by the following points, by performing:

· a review of Arduino and Raspberry Pi based portable laboratories;

· a review and identify the most relevant issues regarding on-off and PID controllers;

· a series of control experiments using the TCLab kit involving both simulation and validation tests;

1.3 Thesis Structure

This thesis is organised in six chapters and a reference list. Following this introduction chapter, the thesis remaining is organised as follows:

· Chapter 2 presents and overview of take-home laboratory control equipment;

· Chapter 3 introduces the TCLab control kit main characteristics addressing both; hardware and programming software; Basic introductory TCLab hands-on experiments are also presented;

· Chapter 4, reviews basic issues regarding industrial controllers design;

· Chapter 5, presents a series of control experiments using the TClab;

· Chapter 6, conclusion and future work.

Chapter 2

Take-home Laboratory Control Equipment: An Overview

The purpose of the present chapter is to make a state-of-the-art review regarding Arduino based control kits. In order to develop a project overview, it must meet several requirements set by the needs of its users. In this case, the users are the students.

2.1 System Concept

It is important to notice that the product has to meet certain standards in order to be successful and so a system concept diagram is presented below.

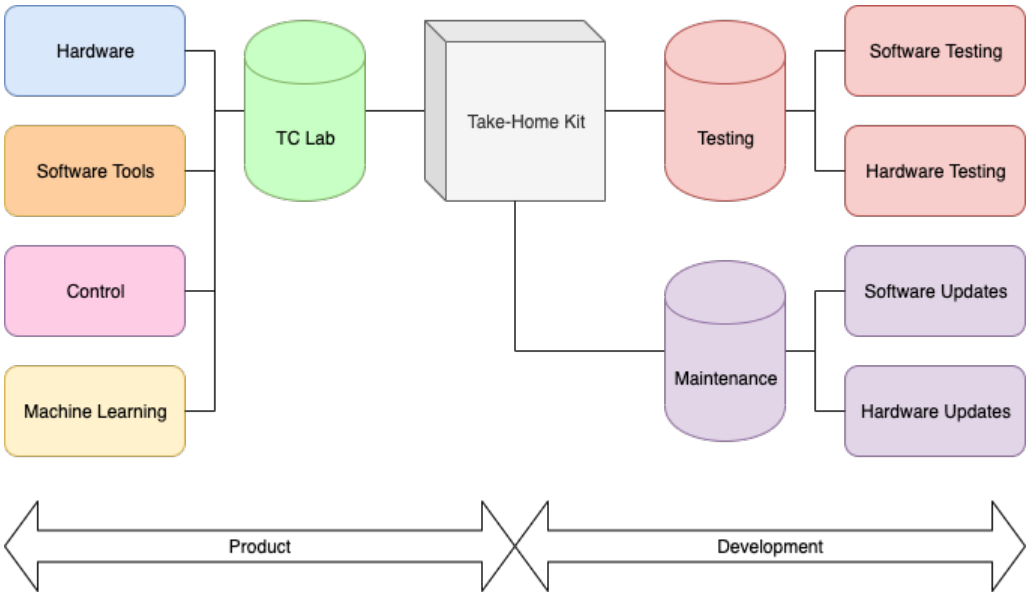

Figure 2.1: System Concept Diagram

The system is based on a TCLab kit. There are two fundamental components of the product being these the Product functionalities and Development Phase.

The kit functionalities are depicted as Product functionalities in the figure. This component is divided in four sectors. The first contact to have with the kit is the hardware part. With further analysis, it is possible to understand the main components and its purpose. Secondly, software tools such as Python, Matlab© and Simulink© are required in order to actuate the termistors, read data surveys and visualise this data.

The temperature control lab is a plug and play Arduino device that allows the teaching of principles of system dynamics and control such as Data Science, Process Dynamics, Programming, Heat Transfer, Machine Learning and Control with real data.

On the other hand, the development phase is composed of Testing and Maintenance phase where the functionalities of the kit are used and studied. The purpose of the testing phase allows students to make their own tests to both software and hardware. This is proven to reinforce the theoretical concepts and thus facilitates the students' learning process.

When speaking of the maintenance phase, this is a key part of the project due to the progress in both software and hardware components allowing a more modern technology in order to improve the learning experience for students.

2.2 System Requirements

System requirements (see 2.1) can be broadly classified as functional requirements, data requirements, quality requirements and constraints. They indicate the minimum and the recommended configuration in order to use the product efficiently.

Table 2.1: Requirements

Description
The system must be user friendly (e.g.USB input port and standard software available)
Code templates should be easy to edit allowing students to implement their tests
The system must be able to support visual data
The system must be able to collect data using it to improve the system model or control tuning
The system must be inexpensive
The system must be easily replaceable
The system should be able to Connect to a standard Windows PC
The system must be robust (Rugged) in order to survive impacts when taking it from place to place
Control data must be displayed graphically

2.3 State-of-the-art Revision

A core obstacle to the inclusion of hardware in engineering curricula is the combination of timetabling and space; laboratories are expensive to provide and thus are limited in general and this in turn means students have limited access. One popular solution to this dichotomy that has proved very popular in recent years is to increase the use of web base materials [16], or more specifically, web based laboratories.

Temperature control devices are an integral component of any general use laboratory and are needed in almost every experimental workflow. With the exception of remote access laboratories which themselves are subject to a number of challenges, remote access activities tend to be based on virtual/mathematical environments and thus are only pseudo-authentic [16].

There are many, complex, real-world systems and processes that directly affect our everyday life. Another innovative approach is to focus on using these systems as the basis for an experiment to better understand the fundamental engineering principles on how they work and how they can be improved. Examples of these systems include automobiles, cooking processes, structures, bio-organic processes, traffic flow, house appliances, human performance and behaviour. A take-home laboratory might focus on the heating and cooling system for a house. It could take data on the operation as well as allow the student to implement control strategies to turn on, for example, a space heater only when really needed in order to save energy [8].

2.3.1 Take Home Helicopter Emulator [1]

This product focused on low cost yet accessible experience that is also portable system aimed for students in order to better support the teaching of advanced control and so, is very similar to Arduino: Temperature Control Laboratory(TCLab). The difference resides in the control application and how to manage its actuators in order to maintain a steady flight. It is used for aerodynamics and bigger emphasis on real-time interactive control.

In this project the parameters being controlled are angular rotation, pitch yawn, tilt, etc. This is done by an array of termistors to read information and actuators to adjust them as new values are calculated. Connections are established trough USB and the system is divided into several modules like data-acquisition, modelling and simulation of multi-physics systems, system identification and multi-variable control and state estimation. Software used is LabVIEW and Matlab©.

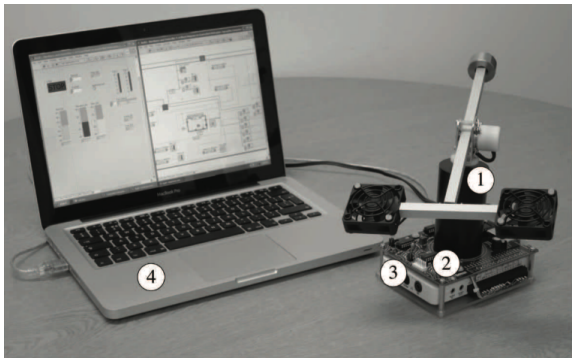

Figure 2.2: Take-Home Helicopter Emulator kit (Extracted from [1])

.

2.3.2 Take-home embedded Control Unit [1]

This kit consists of a development board including a processor and peripherals. It is made with affordability and accessibility in mind. The concept encapsulates students up until the post-graduate level. It is capable of handling remote communications as well as different languages and compilers. Similar to TCLab, the power cable is connected to the device via USB and is simple to setup. The control experiments being tested are more flexible than TCLab's because of the access to accessories that allow different controllers and embedded systems to be developed on the same kit, that is the main advantage of this modular design. The main disadvantage of being modular is the sacrifice to accessibility compared to the built-in heater configuration of other kits. On the other hand, this added complexity permits higher level student teaching applications. The language used is C and Keil μvision software is present.

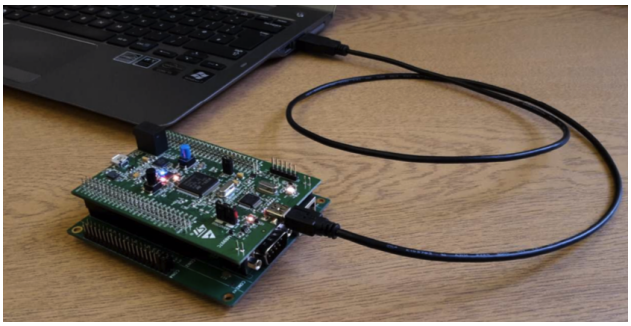

Figure 2.3: Take home embedded systems equipment. (Extracted from [1])

2.3.3 A Refrigerator Energy Monitoring System [2]

This product is unique because, while typical data acquisition systems are usually more expensive, this one is not and functions with both analogue and digital signals.Also, trough the use of a parallel port interface and a modular external connection board, the system becomes portable. It is also possible to control and develop the software from home and affect the refrigerator. This kit was included for the sake of demonstrating the variety of approaches utilised in this medium.

The objective of the learning experience is to teach how to improve the energy efficiency of a refrigerator, with some extra economic considerations added. This starts with a more generic approach and gradually progresses to more in depth calibration. There are also additional features that include alarms.

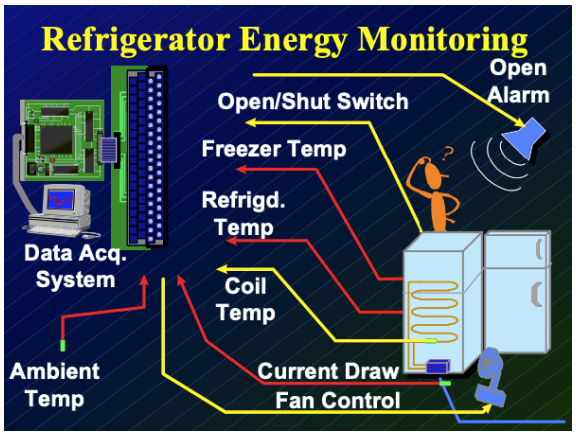

Figure 2.4: Refrigerator Energy Monitoring System diagram (Extracted from [2])

.

2.3.4 Ball and Beam [3]

The system is related to control systems. It is a nonlinear unstable system with two degrees of freedom: the movement of the ball and the beam rotation. It uses termistors and actuators as a means of reading data and adjusting outputs to the controller actualised values. Extra care was taken in the design of the beam to take in account the termistors installed and a DC motor is also part of the design. This kit is Arduino based and reads analogue inputs from an Infrared termistor and the data is then converted to digital by means of binary sampling to be analysed. The Arduino kit is powered by 5 volts when the power cable is connected. One of the more important parameters and considerations is the estimation of the balls centre of mass, without it is difficult to perform accurate control adjustments to the arm.

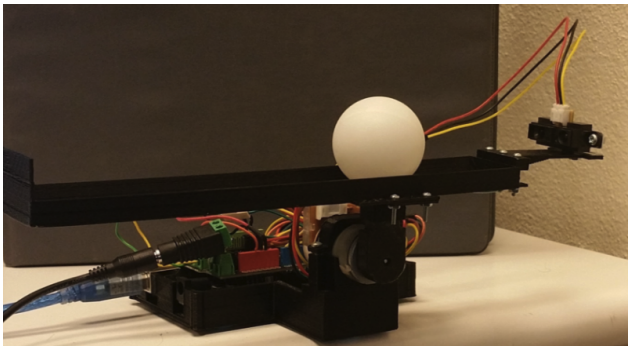

Figure 2.5: Ball and Beam Model (Extracted from [3])

As a conclusion remark to this section, it is possible to create a large number of laboratory projects that can be done at any location and any time desired with the use of inexpensive termistors and system components. Some of the examples can be:

- Motor/controller setup used to teach feedback control techniques;
- Water pumping system;
- Thermodynamic;
- Safety of chemical reactions;

Chapter 3

Temperature Control Laboratory (TCLab)

The Arduino Temperature Control Lab is a modular, portable, and inexpensive solution for hands-on process control learning. Heat output is adjusted by modulating current flow to each of two transistors. Thermistors measure the temperatures. Energy from the transistor output is transferred by conduction and convection to the temperature termistor [17]. The dynamics of heat transfer provide rich opportunities to implement single and multi-variable control systems. The lab is integrated into a small PCB shield which can be mounted to any Arduino or Arduino compatible micro-controller.

In this chapter, the basics of the temperature control laboratory are depicted as well as the used technologies applied throughout the project and a comparison between their performance.

3.1 TCLab Hardware Description

3.1.1 Arduino IDE

The Arduino Integrated Development Environment - or Arduino Software (IDE) uses the C programming language facilitating beginner users to start their own projects. On top of that, it allows access to Arduino Library that is constantly being developed by the open-source community. This software allows the project and design of micro controller based development boards in order to make a project using the Arduino board. It contains a text editor, a message area, a text console, a toolbar with buttons and a series of menus [11]. It connects to the Arduino and Genuino hardware to upload programs and communicate with them. In this case, the Arduino used is **Arduino Leonardo**.

In order to configure the board to the Arduino IDE, the board version must be selected as shown in the following image:

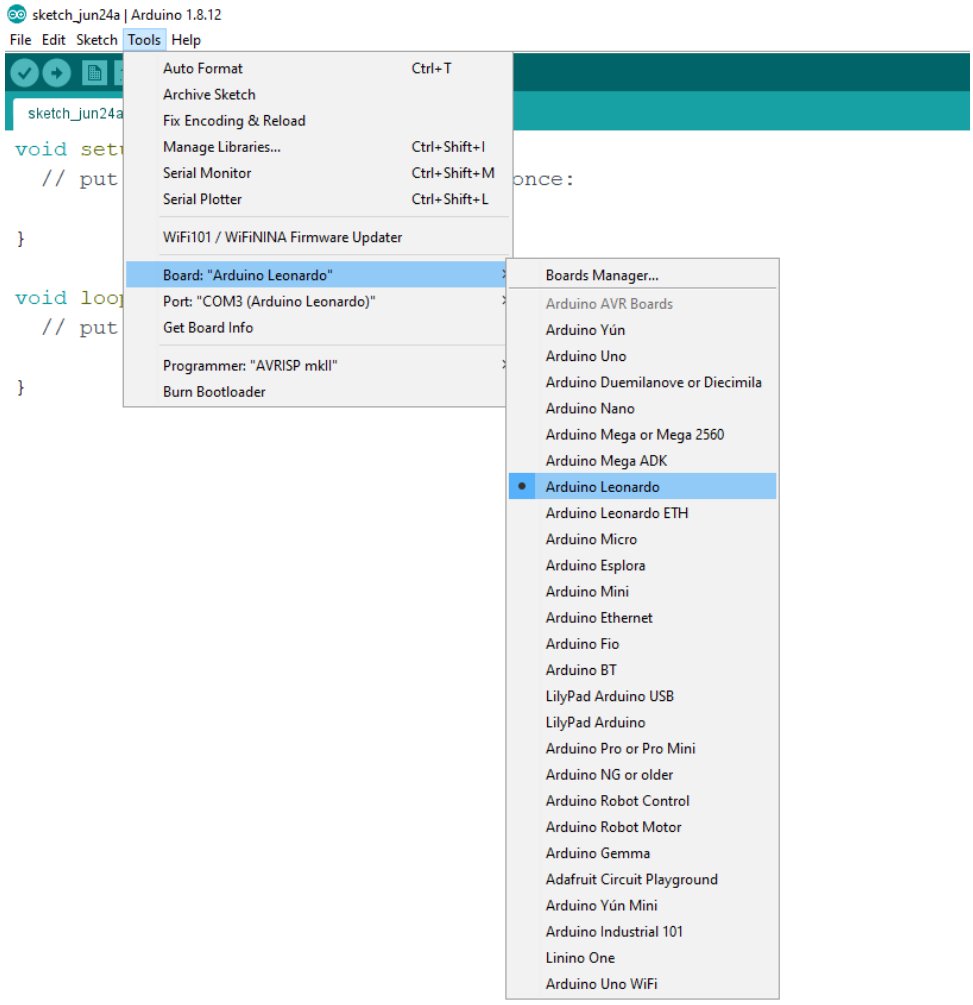

Figure 3.1: Arduino IDE Setup. Extracted from [4]

The COMM port must also be selected according to the user preference.

3.1.2 Overview

TCLab is based in Arduino hardware, more specifically Arduino Leonardo because of the additional USB connection benefits the later option offers in terms of compatibility with a wider range of software. Two cables are supplied and a USB to socket adaptor that when connected to the electrical grid provides the power the device needs to operate. The white cable is the power cable that connects to the adaptor and the round shaped power connector in the Arduino, the second one is blue and has one micro-USB connection on the Arduino side and one normal USB connection for the PC port.

The TCLab is equipped with two heaters and two temperature termistors represented as: h_1, h_2, T_1 and T_2 respectively. In this experiment, only the h_1 heater and corresponding T_1 termistor are used. To control the temperature the voltage applied to the heater varies in a rate defined by the controller. The purpose of the controller is to make the temperature reading in T_1 reach the set-point and maintain the set-point value. Some of the energy is lost by conduction to the exterior in between the space that separates heater from termistor.

Figure 3.2: Pocket-sized process control lab (Extracted from [1])

.

TCLab also includes two transistors in the central upper section that function as heaters with two thermistor termistors coupled to them to measure temperature readings. The TCLab programming language can be Python or Matlab©. An approach consists in using Python to conduct the TCLab control tests and the use Matlab© to data processing and generate graphics.

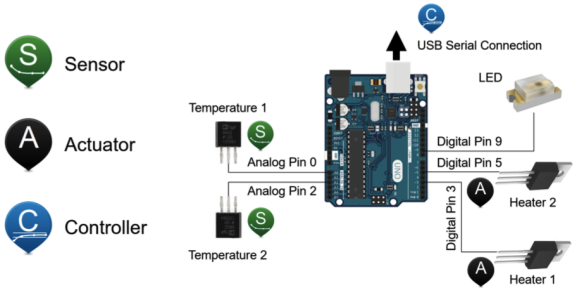

Figure 3.3: Temperature termistors and heater actuators with connections to an Arduino (Extracted from [1])

.

3.2 TCLab Characteristics

To conclude, a final analysis of the pros and cons of the more traditional approach and the one proposed by devices such as TCLab was made and the results are shown in the following table:

Table 3.1: Comparison between traditional *vs* take-home laboratory projects [8]

Issue	**Standard Lab**	**Take-Home Lab**
Learning	– Focused on specific aspect of overall system	– Promotes self learning and independent study – Can focus on real-world systems
Cost	– Upfront costs: High – Space costs: High – Operation Costs: Moderate – Labor costs: High	– Upfront costs: Moderate – Space costs: none – Operation Costs : Moderate – Labor costs: low
Advantages	– Hands-on learning – Complex material available – Instructors to help overcome difficulties	– Hands-on learning – Portable – Independent learning
Disadvantages	– Limited time to complete tasks – Space may not be available – Student scheduling issues	– Need for documentation – PC and software at home – Lower speed data acquisition – Can't teach use of sophisticated test equipment

3.3 TCLab Programming

3.3.1 Python Software

3.3.1.1 Python

Python is a high-level programming language that allows an easy comprehension for beginners. Python has a broad variety of libraries that are very well suited to develop customised machine learning tools [18].

3.3.1.2 Anaconda, Inc.

Anaconda, Inc. is an open-source software that is aimed to be used in data science and machine learning applications. It is a distribution platform that uses both Python and R programming languages and aims to simplify package management and deployment. Package versions are managed by the package management system Anaconda. This Python distribution has an interpreter that runs the project and it also comes bundled with an editor called Spyder[5]. In Figure 3.4 it is shown the main menu of the Anaconda interface. In the top right corner it is possible to select the Spyder editor to write Python code.

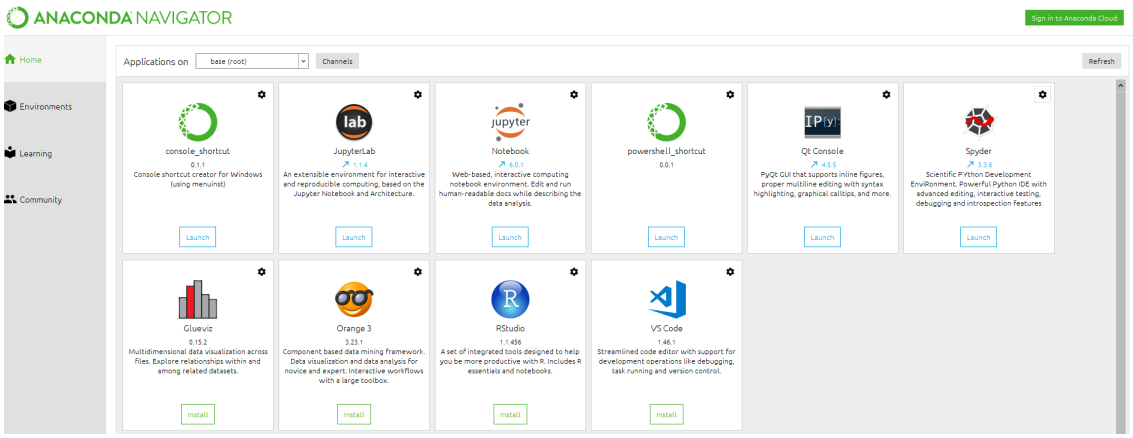

Figure 3.4: Anaconda tools selection window.[5]

3.3.1.3 Python Libraries

This software has the following built-in packages:

Table 3.2: Python Libraries

Description	Python Libraries
TCLab Package	tclab
Scientific Computing	numpy
Get Data Timestamp	time
Interactive Plots	matplotlib.pyplot
Information on the Python Interpreter	sys

The TCLab package includes:

- **Clock-** Python generator for real-time implementation of process control algorithms;
- **Historian-** Python class to log results of a process control experiment;
- **Plotter-** Provides an historian with real-time plotting within a Jupyter notebook(open-source web application that allows the creation and document sharing that contain live code, equations, visualisations and narrative text);
- **TCLabModel-** Embedded model of the temperature control lab for off-line and faster-than-realtime simulation of process control experiments; [19]

3.3.1.4 Setup

In Figure 3.5 a snippet of the Arduino IDE code used to import the necessary TCLab library to the Leonardo kit is shown. To successfully perform the import there are two necessary steps, the compilation and data upload.

```
// constants
const String vers = "1.01";     // version of this firmware
const int baud = 9600;          // serial baud rate
const char sp = ' ';            // command separator
const char nl = '\n';           // command terminator

// pin numbers corresponding to signals on the TC Lab Shield
const int pinT1   = 0;          // T1
const int pinT2   = 2;          // T2
const int pinQ1   = 3;          // Q1
const int pinQ2   = 5;          // Q2
const int pinLED  = 9;          // LED

// global variables
char Buffer[64];                // buffer for parsing serial input
String cmd;                     // command
float pv;                       // pin value
float level;                    // LED level (0-100%)
float Q1 = 0;                   // value written to Q1 pin
float Q2 = 0;                   // value written to Q2 pin
int iwrite = 0;                 // integer value for writing
float dwrite = 0;               // float value for writing
int n = 10;                     // number of samples for each temperature measurement

void parseSerial(void) {
  int ByteCount = Serial.readBytesUntil(nl,Buffer,sizeof(Buffer));
  String read_ = String(Buffer);
```

Figure 3.5: Python Arduino initialisation.

In Figure 3.6 the Python script necessary to complement the TCLab library that was uploaded to the kit is shown. This script acts as the interface that allows the user to make programs using the functions and variables that easily control the heaters and read from the actuators. Additionally the initiations necessary to open and close the serial communications are also performed in this script.

tclab.py

```
import sys
import time
import numpy as np
try:
    import serial
except:
    import pip
    pip.main(['install','pyserial'])
    import serial
from serial.tools import list_ports

class TCLab(object):

    def __init__(self, port=None, baud=9600):
        port = 'COM3' #self.findPort()
        print('Opening connection')
        self.sp = serial.Serial(port=port, baudrate=baud, timeout=2)
        self.sp.flushInput()
        self.sp.flushOutput()
        time.sleep(3)
        print('TCLab connected via Arduino on port ' + port)

    def findPort(self):
        found = False
        for port in list(list_ports.comports()):
            # Arduino Uno
            if port[2].startswith('USB VID:PID=16D0:0613'):
                port = port[0]
                found = True
            # Arduino HDuino
            if port[2].startswith('USB VID:PID=1A86:7523'):
                port = port[0]
                found = True
            # Arduino Leonardo
            if port[2].startswith('USB VID:PID=2341:8036'):
                port = port[0]
                found = True
        if (not found):
            print('Arduino COM port not found')
            print('Please ensure that the USB cable is connected')
            print('--- Printing Serial Ports ---')
            for port in list(serial.tools.list_ports.comports()):
                print(port[0] + ' ' + port[1] + ' ' + port[2])
            print('For Windows:')
            print('  Open device manager, select "Ports (COM & LPT)"')
            print('  Look for COM port of Arduino such as COM4')
            print('For MacOS:')
            print('  Open terminal and type: ls /dev/*.')
            print('  Search for /dev/tty.usbmodem* or /dev/tty.usbserial*. The port number is *.')
            print('For Linux')
            print('  Open terminal and type: ls /dev/tty*')
            print('  Search for /dev/ttyUSB* or /dev/ttyACM*. The port number is *.')
            print('')
            port = input('Input port: ')
            # or hard-code it here
            #port = 'COM6' # for Windows
            #port = '/dev/tty.wchusbserial1410' # for MacOS
        return port
```

Figure 3.6: Python Spyder initialisation

In Figure 3.7, a test using a proportional controller and a fifteen minute experiment with three different steps was performed. In this experiment the main focus isn't the controller performance but the noise observed in the graphic below that illustrates the heater action on the blue $Q1$ line. The observable noise level represented in spikes an irregularities in this line is very low.

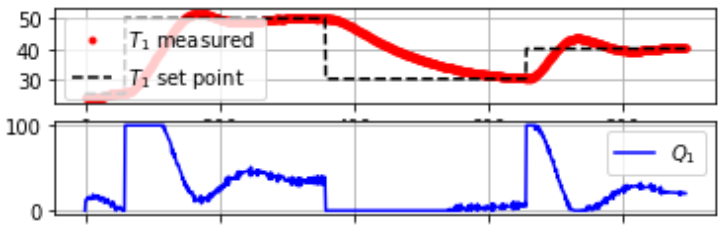

Figure 3.7: Heater test using Python Script

3.3.2 Matlab© and Simulink©

The TCLab emphasises parameter estimation, transient modelling as well the feedback control to maintain temperature and it enables teaching principles of system dynamics and control. In particular, this lab reinforces:

- Dynamic modeling with balance equations;
- The difference between manual and automatic control;
- Step tests to generate dynamic data;
- Fitting dynamic data to a First Order Plus Dead Time (FOPDT) model:
- Obtaining parameters for PID control from standard tuning rules;
- Tuning the PID controller to improve performance;
- Process Control Temperature Lab; [20]

The temperature termistors in TCLab have a multi-variate and dynamic responses. With this, it is possible to use Matlab© and Simulink© in order to obtain a dynamic response to heater changes. In Figure 3.8, the support package needed for the Arduino kit to operate and communicate with Matlab© is shown. This support package is installed in several steps in which the user is requested to provide information such as the Arduino type, number of the communications port and which libraries to select in the installation. The default libraries selected in the program are enough for the installation to be successful.

Figure 3.8: Matlab add-on.[6]

3.3.2.1 Setup

In order to initialise the board in Matlab©, *tclab* command must be called as shown. This command initialises all communications trough the communications port previously selected in the support package installation. In Figure 3.9, the details returned by the command window following a successful initialisation can be observed. The information on display concerns the available pins in the kit, the communication port selected, the type of Arduino board etc.

```
>> tclab
Updating server code on board Leonardo (COM3). This may take a few minutes.
Arduino Leonardo detected.
This device is ready for use with MATLAB Support Package for Arduino Hardware. Get started with examples and other documentation.
To use this device with Simulink, install Simulink Support Package for Arduino Hardware.

  arduino with properties:

                Port: 'COM3'
               Board: 'Leonardo'
       AvailablePins: {'D2-D13', 'A0-A5'}
  AvailableDigitalPins: {'D2-D13', 'A0-A5'}
     AvailablePWMPins: {'D3', 'D5-D6', 'D9-D11', 'D13'}
   AvailableAnalogPins: {'A0-A5', 'D4', 'D6', 'D8-D10', 'D12'}
    AvailableI2CBusIDs: [0]
AvailableSerialPortIDs: [1]
             Libraries: {'I2C', 'SPI', 'Servo'}
Show all properties
```

Figure 3.9: Matlab initialisation

Figure 3.10, presents in the top plot the temperature set-point (blue dashed line) and the temperature measured in transistor 1 (solid red line). The bottom plot presents the actuator heater 1 signal. In experiments conducted in Matlab© this noise level is relatively low, although some spikes are present.

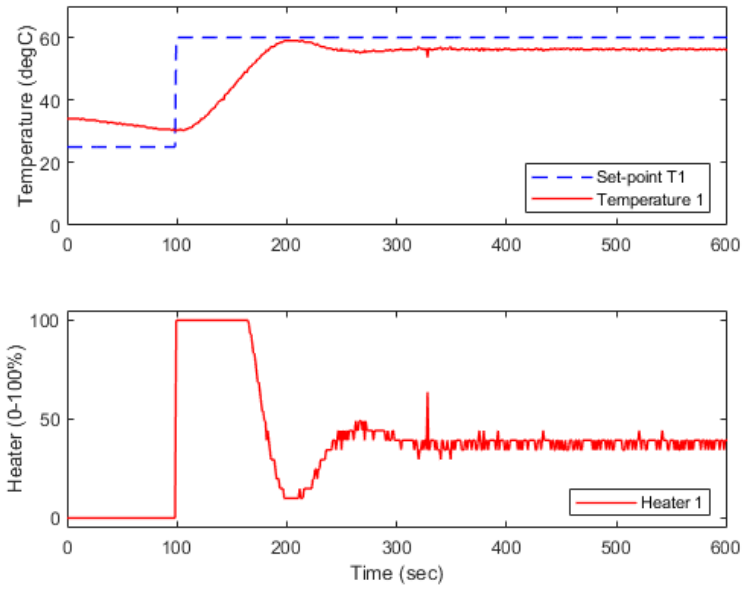

Figure 3.10: Matlab heater test noise example.

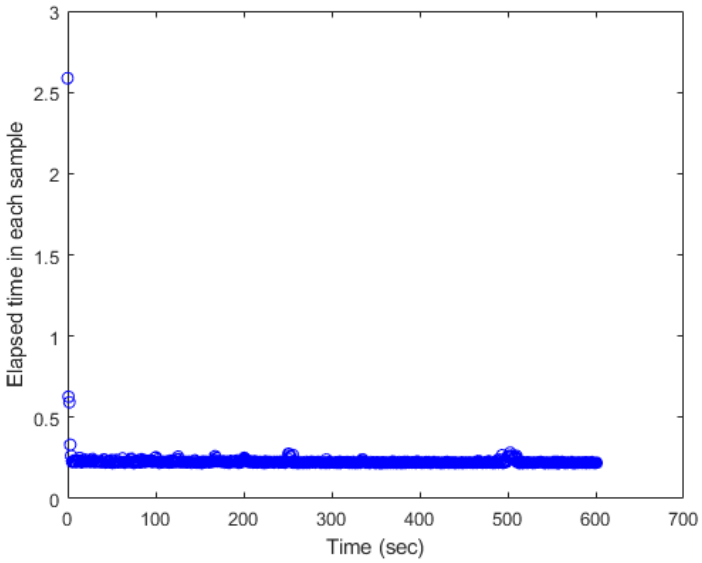

Figure 3.11: Elapsed time measured in every control loop execution in Matlab

3.3.3 GNU Octave

GNU Octave is a software displayed as a high-level programming language aimed for the analytical and numerical computation. This software solves both linear and nonlinear problems numerically and allows performing other analytical experiments using a language compatible with Matlab©. This software is very interesting since it is open-source, presenting a free-to-use alternative to the Matlab© software. Others alternatives may be Scilab and FreeMat.

3.3.3.1 Setup

In Figure 3.12, the Arduino IDE script needed to upload the library interface of the Octave program to the Arduino kit is shown. This is script is similar to the one used in Python and serves the same purpose. The functions that initialise the TCLab kit in the different environments can be found in this reference: [4].

```
#include "settings.h"


#include "LibraryBase.h"

// lib manager / processing
static OctaveArduinoClass octavearduino;

// include the base library
#include "OctaveCoreLibrary.h"
OctaveCoreLibrary lib0(octavearduino);

#ifdef USE_I2C
#include "OctaveI2CLibrary.h"
OctaveI2CLibrary lib1(octavearduino);
#endif

#ifdef USE_SPI
#include "OctaveSPILibrary.h"
OctaveSPILibrary lib2(octavearduino);
#endif

#ifdef USE_SERVO
#include "OctaveServoLibrary.h"
OctaveServoLibrary lib3(octavearduino);
#endif

#ifdef USE_SHIFTREG
#include "OctaveShiftRegisterLibrary.h"
```

Figure 3.12: Octave Arduino initialisation

In Figure 3.13, the sequence of instructions necessary to initiate the Arduino to be programmable in Octave is shown., Before executing this commands the serial communications package for Octave must be installed using the command: pkg load instrument-control. After the successful installation of the instrument control package the default Arduino package for Octave must be installed with the pkg load arduino command. Next, the arduinosetup command brings up a window prompt to Arduino IDE in order to upload the TCLab libraries for Octave to the kit (see Figure 3.12). Command scanForArduinos is optional and returns the port and type of Arduino kit (in this case, Leonardo) if all previous steps were successful. Finally, function tclabsetup octave, the same function as tclab in Figure 3.9 with a few adjustments to be compatible with Octave is called to initialise the variables and functions specific to TCLab.

```
>> pkg load arduino
>> arduinosetup
Running "C:\Program Files (x86)\Arduino\arduino.exe" "C:\Users\BRUNOA~1\AppData\Local\Temp\oct-AlDTUB\octave\octave.ino"
ans = 1
>> scanForArduinos
ans =
{
  [1,1] =

    scalar structure containing the fields:

      port = \\.\COM3
      board = leonardo

}

>> tclabsetup_octave
<object arduino>
```

Figure 3.13: Octave initialisation

In Figure 3.14, the comparison between the elapsed time measured between control loop executions using Octave is notably more inconsistent and presents a higher value in average than the samples taken using Matlab© in Figure 3.11. Octave registers values between 650 milliseconds and 750 milliseconds trough the test execution.

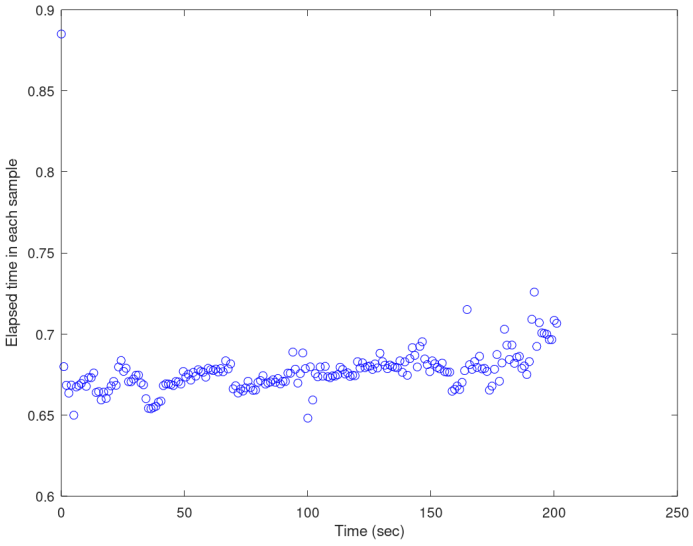

Figure 3.14: Elapsed time measured in every control loop execution in Octave

3.3.4 Discussion

The TCLab kit is compatible with Matlab©, Python, GNU Octave, Java, and any software that can send and receive commands from a serial USB connection. While Matlab© requires the Arduino support package, Octave requires the Arduino package and the installation of the instrument control package. Also, Python requires the TCLab package that can be installed using pip or Arduino IDE to install directly in the board. Python also requires a separate environment to code in, this environment is the Spyder utility located within the Anaconda package. The noise results demonstrate that Python produces the less noise of all platforms. While Matlab© presents consistent and low (200 millisecond) inter-sample time registers it has a slightly worse performance than Python with occasional spikes in noise. Octave is by far the worse performer because it presents a high and more inconsistent inter-sample time register (700 milliseconds) and very high levels of noise without a filter. These levels can be improved with the use of a filter but nevertheless remain worse than the ones present in Matlab© an Python. It is important to note that Matlab© and Python are free and Matlab© has a paid license.

For more information and to access the scripts used throughout this dissertation a website is available in the following link: `http://fe.up.pt/asousa/tclab`.

3.4 TCLab Basic Tests

In this section a couple of tests are made to ensure the basic elements of the termistors and actuators that are part of the Arduino Leonardo kit are in working order by activating the heaters to full capacity for a duration of 10 minutes (600 samples). Then, assuming everything was configured correctly in the software installation and communication test phase, the results will show a graph with two axis temperature and time. The temperature indicated by the termistor is going to rise accordingly throughout the experiment, as well as the program being able to activate one of the LED's that are included in the kit in a given cadence and with controlled intensity.

3.4.1 LED Test

After compiling and running the following program: This program will test if the commands from the software communicate adequately trough the serial USB connection between the PC and the TCLab. If the connection does indeed work there will be no errors or warnings in the console.

```
import tclab
import time

#connect arduino
a =  tclab.TCLab()

#Get Version
print(a.version)

#Turn LED on
print('LED On')
a.LED(100)

#Taper LED off
for i in range(100,-1,-10):
    print('LED Power ' + str(i))
    time.sleep(0.5)
    a.LED(i)

a.close()
```

Figure 3.15: TCLab LED test Python Script

It is expected to obtain the display presented in the print-screen in Figure 3.16:

```
Opening connection
TCLab connected via Arduino on port COM3
<bound method TCLab.version of <tclab.TCLab object at 0x00000239C6624448>>
LED On
LED Power 100
LED Power 90
LED Power 80
LED Power 70
LED Power 60
LED Power 50
LED Power 40
LED Power 30
LED Power 20
LED Power 10
LED Power 0
Arduino disconnected successfully
```

Figure 3.16: LED test command window

The Arduino must be disconnected in the end of a program run in every program including Python in order to make the serial connection available again to other programs when switching between them for comparison or convenience purposes.

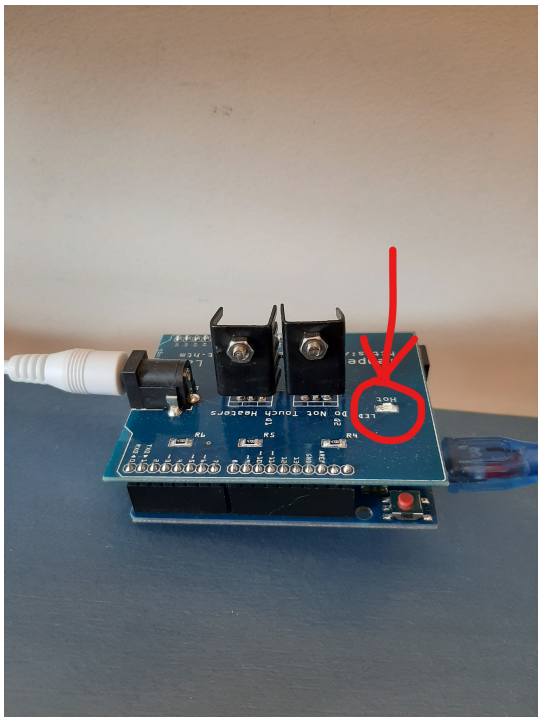

Figure 3.17: Arduino LED OFF

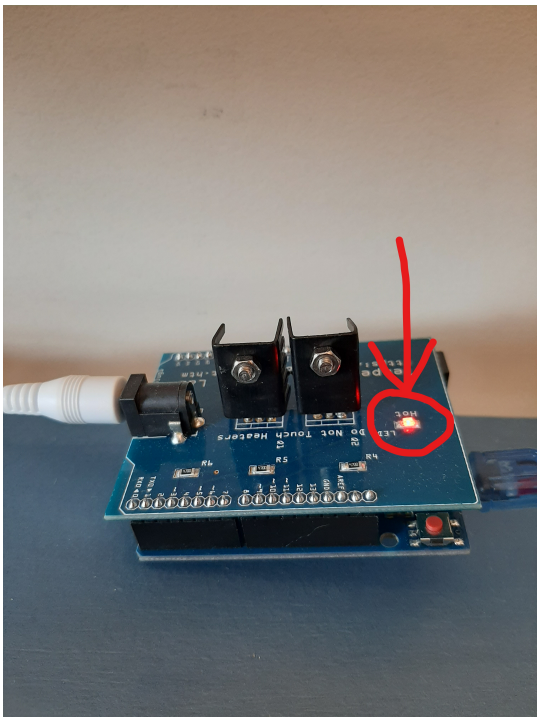

Figure 3.18: Arduino LED ON

After running the program the LED is going to light up, first at full intensity (100%) and then decreasing by 10% every second until shutting off completely. In Figures 3.17 and 3.18, two images demonstrating which LED is being activated in the kit,located in the top right corner.

3.4.2 Heater Test

Moving on to the heater tests, in this work only one of them is going to be activated at once. But the code can be altered easily as to include the second heater in the test. The reasoning behind this choice comes from the fact that only one of the heaters is going to be used in the control experiments ahead.

In Figure 3.19, it is presented an extract of the Python code to test the kits actuators. It is only an extract due to the original code being too large to properly show here as well as the most relevant lines not related to Pythons' specific methods of displaying graphical information are revealed. More importantly, the reference added makes the original code available for any one who is interested to consult.

```
import tclab
import numpy as np
import time
import matplotlib.pyplot as plt

# Connect to Arduino
a = tclab.TCLab()

# Get Version
print(a.version)

# Turn LED on
print('LED On')
a.LED(100)

# Run time in minutes
run_time = 10.0

# Number of cycles
loops = int(60.0*run_time)
tm = np.zeros(loops)

# Temperature (K)
Tsp1 = np.ones(loops) * 23.0 # set point (degC)
T1 = np.ones(loops) * a.T1 # measured T (degC)

Tsp2 = np.ones(loops) * 23.0 # set point (degC)
T2 = np.ones(loops) * a.T2 # measured T (degC)

# step test (0 - 100%)
Q1 = np.ones(loops) * 0.0
Q2 = np.ones(loops) * 0.0
Q1[10:] = 80.0

print('Running Main Loop. Ctrl-C to end.')
print('  Time   Q1     Q2    T1     T2')
print('{:6.1f} {:6.2f} {:6.2f} {:6.2f} {:6.2f}'.format(tm[0], \
                                                       Q1[0], \
                                                       Q2[0], \
                                                       T1[0], \
                                                       T2[0]))

# Create plot
plt.figure(figsize=(10,7))
plt.ion()
plt.show()
```

Figure 3.19: TCLab heater Test experiment Python Script. Extracted from [7]

Figure 3.20 result displays the expected rise in temperature from the ambient around 20 degrees to a value around 80 degrees that is close to the maximum temperature the heater can achieve at the end of 10 minutes of activity.

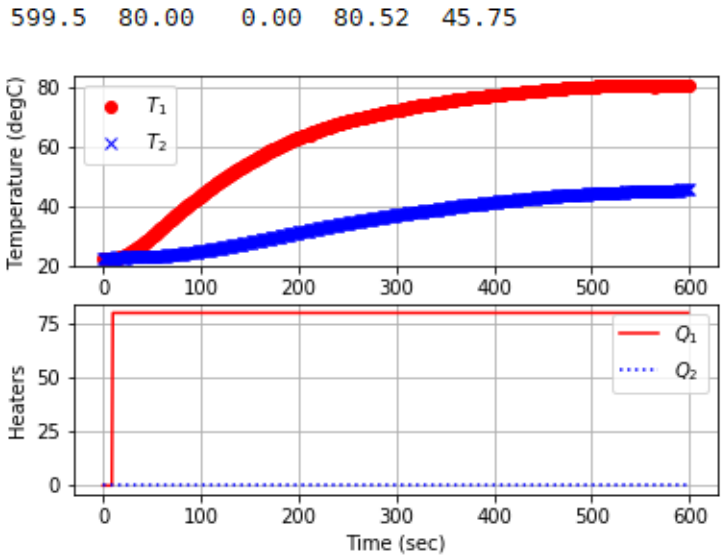

Figure 3.20: TCLab heater test in Python. The top plot presents both measured temperature for transistor 1 and transistor 2. The bottom plot presents the heater signals for transistors 1 and 2.

Chapter 4

Industrial Control Design: Basic Issues

In this chapter introductory concepts regarding the mostly used industrial controllers are presented. The order in which they are presented follows from a sequence that can be used also to teach these concepts. It starts with On-Off control, which does not require any model to design the controller. Then it introduces identification methods for estimating parameters of a first order plus time delay model, which bases the PID control modes design methods. It briefly describes PID control and some of its drawbacks.

4.1 ON-OFF Control

4.1.1 Simple ON-OFF

The first control method addressed is by far the simplest to implement and as it does not require obtaining the system model as well as tuning of the controller parameters. This control method is fairly easy to program in both the version without hysteresis and with. The script in Figure 4.1 is written and run in Matlab©. The main focus upon writing this code is the control loop itself. The control loop is the section directly below the comment that shows "On-Off Control without hysteresis". This section of Matlab© Script determines the controller behaviour to be able to control the heaters in a way to maintain the temperature read by the termistor (i.e. temperature in Transistor 1) in a set level by increasing or reducing the intensity at which they operate. In this case, it is a simple if clause that activates the heaters to 100% of the measured temperature if the termistor detect that it sits below the set-point or 0% if it sits above. In this version of the controller without a hysteresis band there is no buffer zone of action in the Matlab© Script. Figure 4.1 presents an on-off control test obtained with TCLab. The top plot presents the set-point for temperature in transistor 1 (blue line) and the measured temperature in transistor 1 (red line). In the bot plot it is presented the actuator signal for heater 1 (red line) and for heater 1 (dashed blue line).

In Figure 4.2 it is possible to assume, even without analysing performance metrics in detail, that the performance of this type of controller is far from ideal with a considerable overshoot

```
% set point (degC)
tl_sp = 35;

for i = 1:300
    tic;
    % read temperatures
    t1 = T1C();
    t2 = T2C();
    % LED brightness
    brightness1 = (t1 - 30)/50.0;  % <30degC off, >100degC full brightness
    brightness2 = (t2 - 30)/50.0;  % <30degC off, >100degC full brightness
    brightness = max(brightness1,brightness2);
    brightness = max(0,min(1,brightness)); % limit 0-1
    led(brightness);

    % On Off Control
    if t1>t1_sp
       ht1 = 0;
    else
       ht1 = 100;
```

Figure 4.1: Matlab© Script for ON-OFF Controller with Hysteresis Band.

and without ever settling completely in an acceptable 2% region around the set-point. The rising time is similar to other controllers as the heaters are turned on in full right as the set-point is set to a temperature higher than the ambient temperature (temperature measured at t_0 s). In this variation of the on-off controller the hysteresis band is absent and the effect of this can be observed in the graphical representation of the heaters levels below the graphic depicting the function of temperature and time commutation.

In Figure 4.2, it becomes evident that small variations in temperature readings due to hardware limitations around the set-point area can cause an undesirable switching in the actuators when a single reading goes below or above the set-point contradicting the ascending or descending tendency. As a result, instead of a clean graphic, various adjacent lines of quick commutations can be seen next to the ones that are useful and necessary that are separated by several tens of seconds.

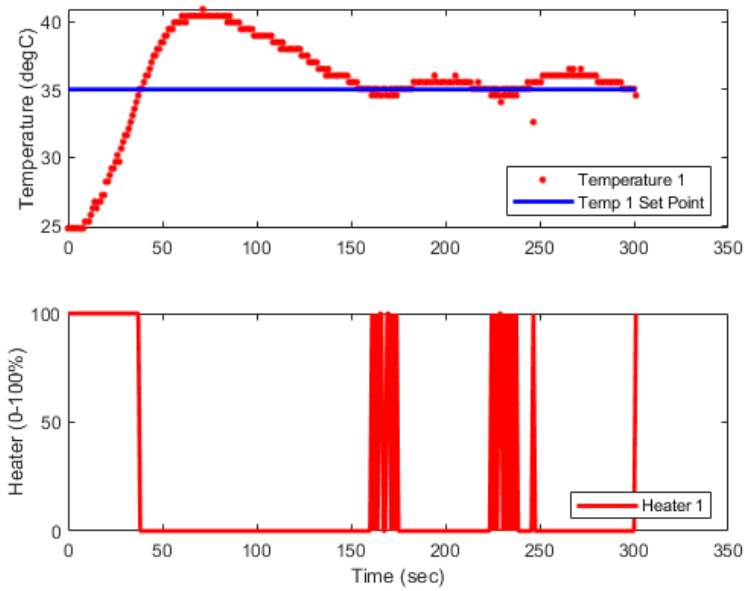

Figure 4.2: ON-OFF without hysteresis TCLab test. The top plot presents both the set-point and measured temperature for transistor 1. The bottom plot presents the heater signals for transistor 1.

4.1.2 ON-OFF with hysteresis

In Figure 4.3 the control loop changes the condition in order to include a hysteresis band. In this case the band was a defined as an asymmetrical band with 0.1 degrees amplitude above the set-point and 0.2 degrees amplitude below the set-point for a total amplitude of 0.3 degrees. This value is arbitrary and based on experiments with the aim to reduce unnecessary commutations in the heater without delaying the controller response to cross the set-point excessively. The anomalies that generate discontinuities in the temperature readings are due to inconsistent sampling times in the software being used. The differences between how prevalent this issue is in each platform will be examined at a later chapter.

In Figure 4.4 the controller performance using the hysteresis band is identical to the one analysed in Figure 4.2. The difference in performance is found in the absence of unhelpful and possibly harmful unintended commutations in the actuation of the heater when the temperature curve passes around the area of the set-point line. This was achieved by adding the hysteresis band as a buffer of sorts that filters the odd spike in temperature readings and prevents the controller from acting based on that information as easily.

```
% set point (degC)
t1_sp = 35;

for i = 1:300
    tic;

    % read temperatures
    t1 = T1C();
    t2 = T2C();

    % LED brightness
    brightness1 = (t1 - 30)/50.0;  % <30degC off, >100degC full brightness
    brightness2 = (t2 - 30)/50.0;  % <30degC off, >100degC full brightness
    brightness = max(brightness1,brightness2);
    brightness = max(0,min(1,brightness)); % limit 0-1
    led(brightness);

    % On Off Control with hysteresis
    if t1-t1_sp>0.1
         ht1=0;
    end
    if t1_sp-t1>0.2
         ht1=100;
    end
     h1(ht1);
```

Figure 4.3: Matlab© Script for ON-OFF Controller with Hysteresis Band.

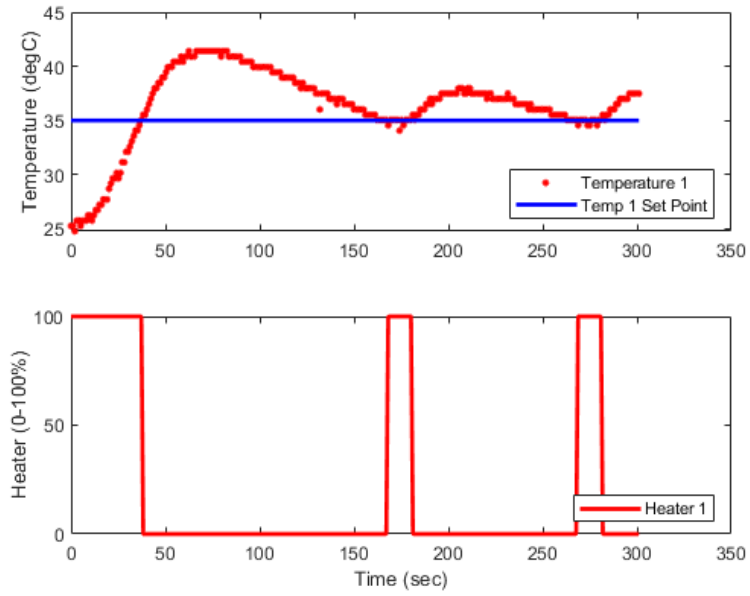

Figure 4.4: ON-OFF with hysteresis band TCLab test. The top plot presents both the set-point and measured temperature for transistor 1. The bottom plot presents the heater signals for transistor 1.

4.2 Process Modelling

A model that can be used to represent a wide range of process dynamics for the most common processes is the first order plus time delay model represented by transfer function (4.1),), with K representing the dc gain, T the dominant time constant and L the pure time delay.

$$G_p(s) = K\frac{1}{(1+sT)}e^{-sL} \tag{4.1}$$

There are many methods to identify the FOPTD model parameters from the open-loop step response. The method that is going to be deployed in this work is the 2-point method. This classical method is based in the determination of the values in two separate specific points in the graphical line representing the systems behaviour, at 35,2% (t_{3_5}) and 85,3%(t_{8_5}) of the final value. Then, the model parameter calculations are made using the following equations:

$$K = \frac{y_{ss} - y_0}{u_{ss} - u_0} \tag{4.2}$$

$$T = 0.67(t_{8_5} - t_{3_5}) \tag{4.3}$$

$$L = (1.3 \times t_{8_5} - 0.29 \times t_{3_5}) \tag{4.4}$$

Alternatively a optimisation method can be used to obtain a refined model.

4.3 Proportional Control

As stating by its name, the proportional control signal is directly proportional to the error signal (see Fig. 4.1). This proportionality is only valid within the linear limits imposed by the actuator.

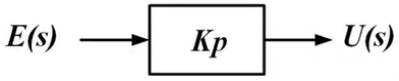

Figure 4.5: Proportional transfer function

There are a huge amount of tuning rules to design PID controllers. In this work the Cohen and Coon method is going to be used for two reasons: i) after the well-known Ziegler-Nichols tuning rules [21] introduced in (1942) the Cohen-Coon [22] become one of the most used tuning rules ii) it was found they perform well to control TCLab [23]. The Cohen-Coon used to evaluate the proportional gain is the following:

$$K_p = \frac{1}{K} \times \frac{T}{L} \left(1 + \frac{L}{3 \times T} \right) \tag{4.5}$$

The major drawback of proportional control is the steady-state error that this method cannot eliminate in practice. Furthermore, it is necessary to limit the control output to an interval between zero and one hundred percent that represents physical limitations present in the TCLab systems actuators.

$$u(kT) = Kp \times e(kT) \tag{4.6}$$

With T representing the sampling interval.

4.4 Proportional and Integrative Control

In integrative control the integral accompanies and registers the accumulation of the error instead of just applying a fixed proportional gain to it and thus adjusting the response of the system to said error in a way that eliminates the steady state error present in the previous proportional controller. Basically, the system now has a memory where it "stores" the error left to correct.

The problem with this method is that when the value of the errors integer demands more response from the actuator than it can provide because its saturated, it accumulates said error in a way that causes unwanted overshoot. This effect is called windup. To eliminate this overshoot an anti-windup technique that resets the integrated error value when it detects that the actuator is saturated. The PI controller transfer function can be represented in the Laplace domain by (4.7) with K_p representing the proportional gain and T_i representing the integral time constant.

$$G_c(s) = K_p\left(1 + \frac{1}{sT_i}\right) \tag{4.7}$$

An Arduino software and Python language it is necessary to deal with digital signals that are converted from the analogue signal the termistor generates. The integrative of the error is approximated from the continuous time domain to the discrete domain using the backward rule, as it is expressed in the following expression, representing the sum of rectangular areas sum:

$$\frac{1}{T_i}\sum_{i}^{k} e(iT) \tag{4.8}$$

Additionally, the Cohen-Coon expressions used to evaluate the proportional gain and the integral time constant are the following:

$$K_p = \frac{1}{k} \times \frac{T}{L}\left(1.9 + \frac{L}{12T}\right) \tag{4.9}$$

$$T_i = L\frac{(30T - 3L)}{(9T + 20L)} \tag{4.10}$$

4.5 Proportional and Derivative Control

In derivative control the system reacts to the derivative of the error meaning it uses the rate of variation of the error. If its increasing fast it will react with a fast increase in actuation if it reducing slowly it will respond with a slow minor decrease in actuation and so on. This method is both advantageous because it reacts quickly to all alterations that occur in the systems curve but it is also affected negatively for being extra sensitive when the input curve applied to the system is a step for example, that would cause an infinite value derivative in theory. This extra sensitivity also means that this form of controller is more affected by noise.

The solve the first problem mentioned, sudden abrupt over reactions, the controllers derivative is applied to the output of the system where the sudden shifts in magnitude have been softened by the transfer function.The PD controller transfer function can be represented in the Laplace domain by (4.9) with K_p representing the proportional gain and T_d representing the derivative time constant.

$$G_c(s) = K_p(1 + sT_d) \tag{4.11}$$

In Arduino software and Python language it is necessary to deal with digital signals that are converted from the analogue signal the termistor generates. To accomplish this, the derivatives in discrete time domain may be represented as a difference between two values of temperature divided by the time interval that sets them apart, using the mathematical expression:

$$\left.\frac{de(t)}{dt}\right|_{t=kT} = \frac{e(kT) - e((k-1)T)}{T} \tag{4.12}$$

Additionally, the Cohen-Coon expressions used to evaluate the proportional gain and the derivative time constant are the following:

$$K_p = \frac{1}{K} \times \frac{T}{L}\left(1.25 + \frac{L}{6T}\right) \tag{4.13}$$

$$T_d = L\left(\frac{6T - 2L}{22T + 3L}\right) \tag{4.14}$$

4.6 Proportional, Integrative and Derivative Control

Finally, the best results can be obtained when all three control modes are joined into the proportional, derivative an integrative controller. Keeping the physical limitation defined by the actuator limits, the anti-windup system from the integrator and applying the derivative to the output of the system all problems encountered before, steady-state error, windup and derivative kick are dealt with and we are left with the most satisfactory method of control of the previous shown. The PID controller transfer function can be represented in the Laplace domain by (4.11) with K_p representing the proportional gain, T_i representing the integral time constant and T_d representing the derivative time constant.

$$G_c(s) = K_p\left(1 + \frac{1}{sT_i} + sT_d\right) \tag{4.15}$$

The following equation (4.12) corresponds to a continuous analogue PID controller output, before being sampled to convert into a discrete analogue signal K_p representing the proportional gain, T_i representing the integral time constant and T_d representing the derivative time constant.

$$u(t) = K_p\left(e(t) + \frac{1}{T_i}\int e(t)dt + T_d\frac{de(t)}{dt}\right) \tag{4.16}$$

Additionally, the Cohen-Coon expressions used to evaluate the proportional gain, the derivative time constant and the integral time constant are the following:

$$K_p = \frac{1}{K} \times \frac{T}{L}\left(\frac{4}{3} + \frac{L}{4T}\right) \tag{4.17}$$

$$T_d = L\left(\frac{4}{11 + 2(L/T)}\right) \tag{4.18}$$

$$T_i = L\left(\frac{32 + 6(L/T)}{13 + 8(L/T)}\right) \tag{4.19}$$

Chapter 5

Control Experiments

In this chapter the theoretical concepts explored in chapter 4 and the hardware and software setup instructions that are presented in chapter 3 in conjunction with the familiarity with the TCLab laboratory kit acquired in chapter 2, are applied to TCLab to fulfil the objectives proposed in chapter 1.

Several control methods are explored and put into practical experiments, from the most simple to the more complex traditional control simulations, starting with on-off control and ending with the various tuning options to the PID controller with variations of proportional derivative and integrative controllers. The experimental results are shown in graphical form, along with snippets of the more relevant part of the code used to execute them using the TCLab. The software tools including all of the presented, Matlab©, Python and Octave are included.The performance of the different software used is approached when relevant to the experiment at hand, in this case the application of noise filters. To support the experimental findings, Simulink© is used to run simulations in order to make comparisons with the TCLab tests.

Finally, the methods are compared based on the advantages and disadvantages of each in a qualitative analysis and the performance indices chosen, IAE, ITAE and TV serve to make a more quantitative numeric comparison.

5.1 First order plus time delay model identification

5.1.1 Two-point Method

In Figure 5.1 the two points represented at 35.2% and 85.3% of the open-loop response are highlighted. With the y axis and x axis values corresponding to these two points and the equations presented in sections 4.3, 4.4 and 4.5, it is possible to evaluate the gain K, time constant T and time delay L parameters which define the FOPTD model. These values vary depending on the individual kit and ambient setup. So it is important that, in order to achieve a more accurate model, the experimental step test is performed when these conditions change significantly. The step response obtained for the temperature measured in transistor 1 presented in Figure 5.1 was obtained by applying the step input presented in Figure 3.20.

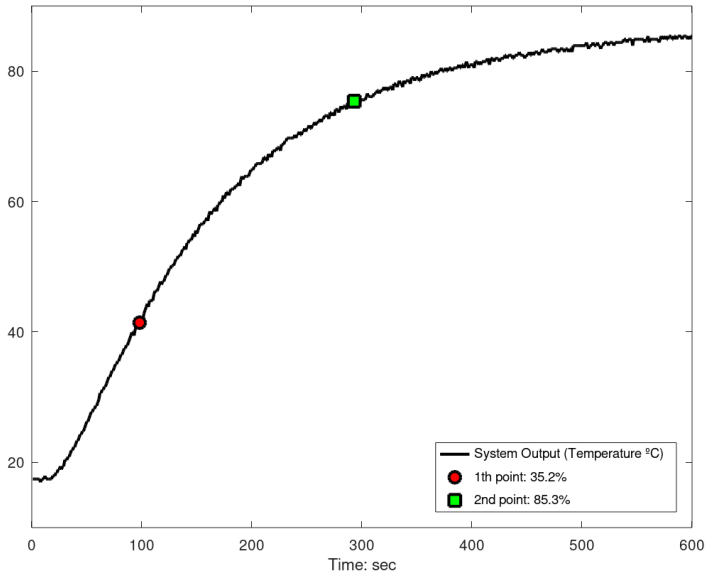

Figure 5.1: Open-loop step response obtained with TCLab. Two-points are outlined corresponding 35.2% and 85.3% of the final steady-state output values.

5.2 Proportional Control

Figure 5.2 presents an example of a digital discrete implementation of proportional control written in Octave.

```
ek = SP_T1(i)-T1; #Error calculation
Pk = Kp*ek;       #Proportional contribution
Q1 = Pk;          #Output calculation

if Q1>100         #Heater saturation
   Q1=100;
 end
 if Q1<0
    Q1=0;
 end

 h1(Q1);          #Heater activation
```

Figure 5.2: Proportional Controller Octave Script.

The first code line calculates the error in the output, meaning that the error is the difference between the set-point and the temperature registered by the termistor at every sampling time. Depending on the program and hardware being used, a discrete sample is taken at an irregular interval that should be theoretically constant but is limited by practical limitations. The second line determines the influence of the proportional component on the controller output. In this case, the proportional component determines the heater output because it is the only one being used.

In Figure 5.3, it is possible to observe that, with this configuration, there is no overshoot present as the temperature never reaches the set-point. As expected, there is also a steady-state error. This type of controller is more advanced than the on-off configuration because it can actuate the heaters on a scale from 0% to 100% and all values in between. The proportional gain obtained using the Cohen-Coon tuning rule was Kp=9.99.

Table 5.1: FOPTD Model Parameters

K	T(s)	L(s)
0.88	18.49	155.90

With this parameters the system expression becomes as follows:

$$G_p(s) = \frac{0.88}{155.9s+1} \times e^{-18.49s} \tag{5.1}$$

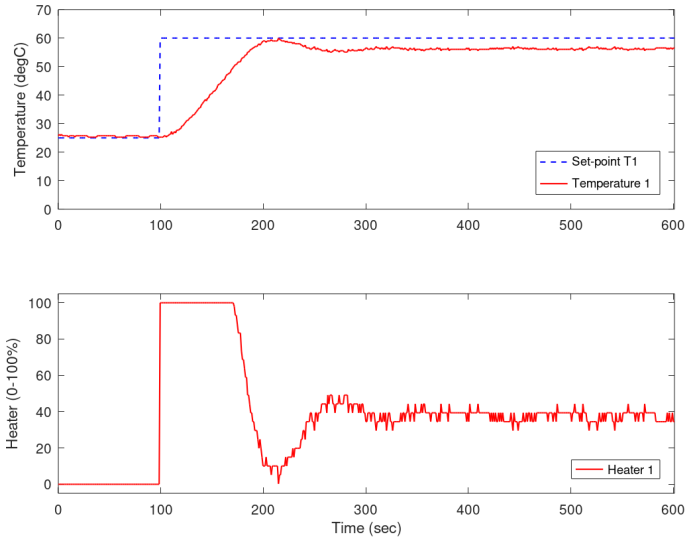

Figure 5.3: TCLab proportional control test obtained with Octave.The top plot presents both the set-point and measured temperature for transistor 1. The bottom plot presents the heater signals for transistor 1.

In order to validate the experimental results, a simulation is needed. In Figure 5.4, the control scheme is presented with different blocks being used to construct the system dynamic, gain, saturation (physical limits of the actuator), steps that define the set-point, functions that represent the system behaviour in the Laplace domain, a block to set the ambient temperature and a block to introduce system time delay.

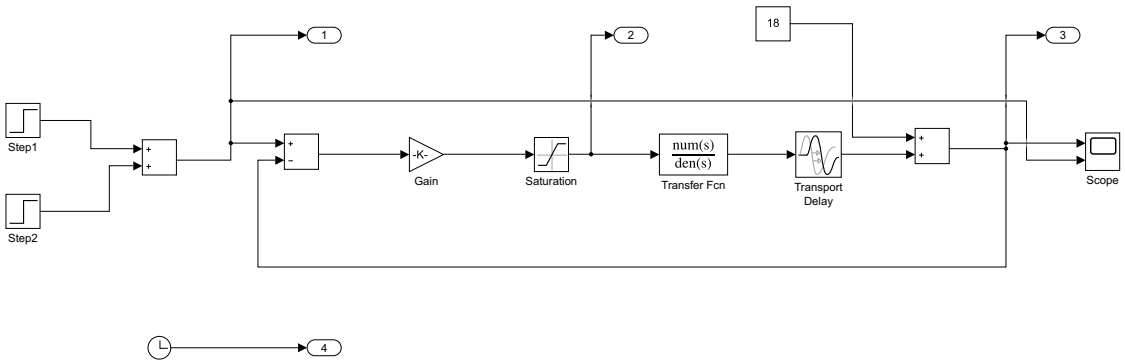

Figure 5.4: Simulink© Model for Proportional Control. Two steps are applied to the system: step 1 at t=0s with amplitude 25 degrees and step 2 at t=100s with amplitude 60 degrees.

The actuation, as the name implies, is directly proportional to the error times a constant Kp that needs to be calibrated depending on the specific application and system dynamic. This makes this type of controller overall more effective at maintaining at least an approximate value to the set-point and especially without constant fluctuations in temperature.

Table 5.2: Performance Indices Table: Proportional Controller

Steady-State Error (%)	IAE	ITAE	TV
5.0	$4.378e+03$	$1.054e+06$	$2.821e+04$

In 5.5, the graphic resulted from the simulation with the same set-points and running time, as well as a sampling time of 1 second that is constant in the simulation and never exceeded in the practical experiment. Therefore, the result is a good approximation of the the practical experiment in all aspects except for the excess oscillation and greater overshoot in the first step.

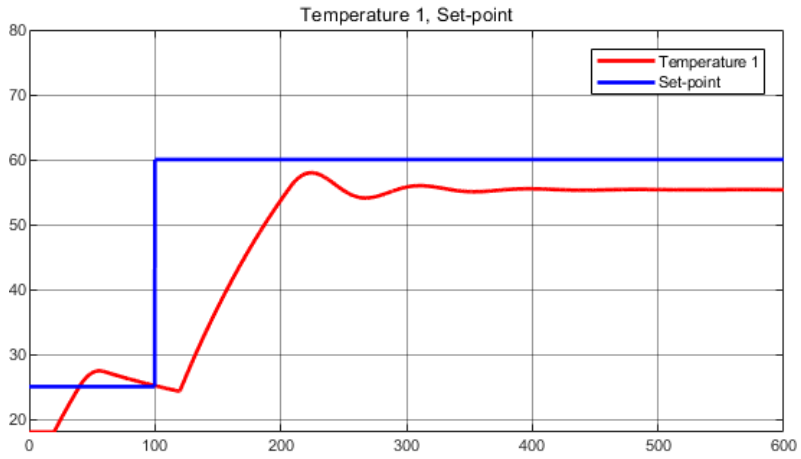

Figure 5.5: Simulated step response obtained with model presented in Figure 5.4. Proportional control with Kp=9.99.

5.3 Proportional and Derivative Control

In Figure 5.6 the proportional controller is given better reaction time by adding the derivative resulting in a proportional and derivative controller (PD). In order to calculate this new position the error value is used again to calculate the derivative contribution.

```
ek = SP_Tl(i)-Tl; #Error calculation
Pk = Kp*ek;       #Proportional contribution
dek =(ek-ek_1)/T; #Error derivative
Dk = Kp*Td*dek;   #Derivative contribution
Dk=(Tf/(Tf+T))*Dk_1 + (T/(Tf+T))*Dk; #Filter
Ql = Pk+Dk;       #Output calculation

if Ql>100         #Heater saturation
  Ql=100;
end
if Ql<0
  Ql=0;
end

hl(Ql);           #Heater activation
ek_1 = ek;        #Previous sample error
Dk_1=Dk;          #Previous sample derivative contribution
```

Figure 5.6: Proportional Derivative Octave Script.

To start, the error is divided by the sampling time, in this case one second, and this value is then multiplied by the derivative constant Kp and the derivative time constant Td, Normally this would be enough to determine the contribution D_k and add it to the proportional P_k to find the value of the heater actuator and apply the proportional derivative controller. In this case, an extra line is added to filter the signal before adding it to the output and this is done by setting a constant T_f, that is worth five in this example, saving the past value of D_k as D_{k1} and executing the operation shown in the line commented as Filter. This is done because the downside to the derivative controllers' added responsiveness is an exacerbated sensitivity to noise.

In Figure (5.7), the effects of not having a filter with a derivative component present in the controller are visible in the graphic representing the actuator's level. The obvious and frequent spikes in heater level are present even in as we enter the steady-state portion of the experiment.

Performance wise, the difference between the basic P controller and the PD control are in the faster reaction time and thus rising time that the derivative portion offers, the specific data will be in the end of the chapter in table 5.11.

Table 5.3: Proportional gain and Derivative time constant values

Kp	Td(s)
12.21	4.77

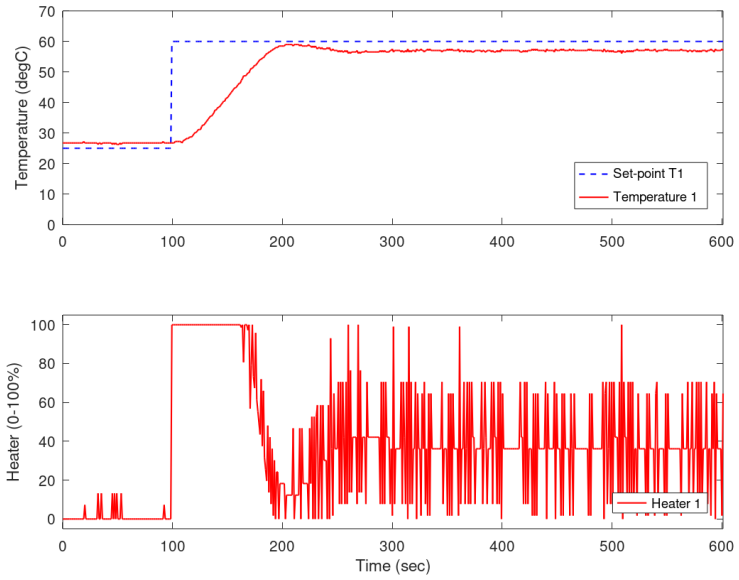

Figure 5.7: TCLab proportional derivative control with no filter test obtained with Octave.The top plot presents both the set-point and measured temperature for transistor 1. The bottom plot presents the heater signals for transistor 1.

In Figure, 5.8 the elapsed time measured in every control loop execution is scattered somewhere between 0,66 seconds and 0,78 seconds never violating the 1 second sampling time. Nevertheless, the frequent and abrupt fluctuations present help generate further noise in the actuator's actions.

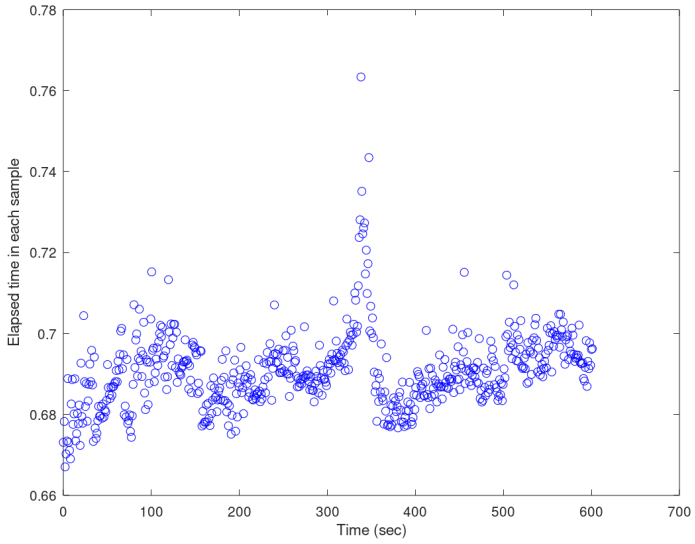

Figure 5.8: Elapsed time measured in every PD control loop execution in Octave.

With the filter equipped in the Octave Script, the heater graphic is much cleaner and the oscillations that previously ranged from anywhere between 0 to 100 % heater level where mostly dealt with leaving some residue of much reduced and less noticeable noise behind. This can be observed in Figure 5.9. Despite having only a minor effect on performance metrics based on overshoot, rising time and steady-state error, the energy consumption of the heater and the wear that it suffers decrease significantly.

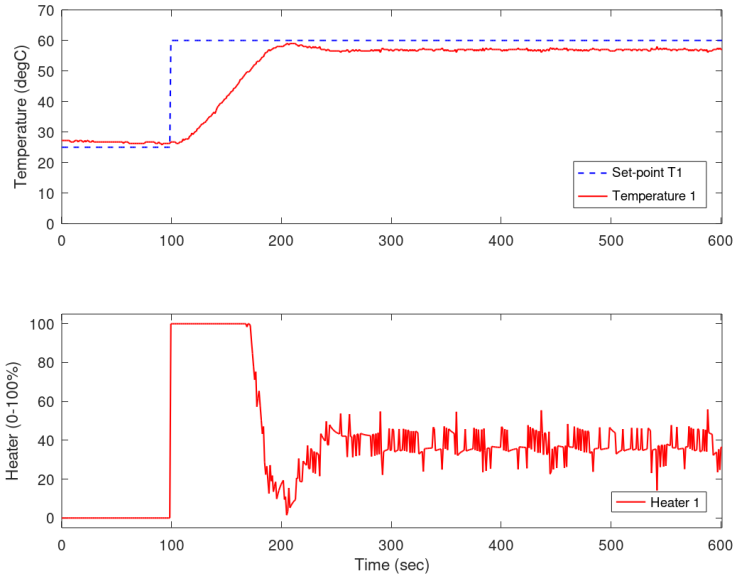

Figure 5.9: TCLab proportional derivative control with filter test obtained with Octave.The top plot presents both the set-point and measured temperature for transistor 1. The bottom plot presents the heater signals for transistor 1.

Table 5.4: Performance Indices Table: Proportional Derivative Controller

Steady-State Error (%)	IAE	ITAE	TV
5.0	$4.386e+03$	$1.060e+06$	$2.813e+04$

In Figure 5.10, elapsed time measured in this experiment grouped than in the example with no filter of Figure 5.8 only registering much smaller and infrequent variations in sample time in a range of 0.65 to 0.75 with the exception of the first sample that is considered an outlier at 0.9 seconds.

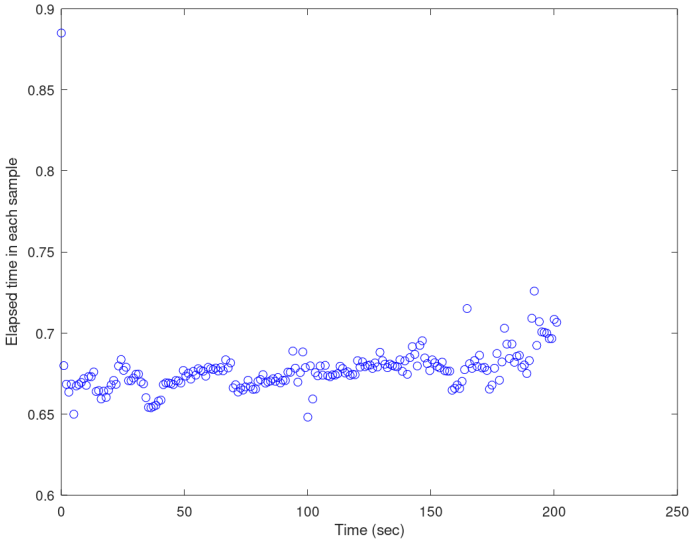

Figure 5.10: Elapsed time measured in every PD control loop execution in Octave with filter.

In Figure 5.11, the second undesirable consequence of adding a derivative component is shown. The derivative-kick effect is when a sudden and abrupt change in the set-point results in an uncontrolled spike in the actuator values. In this example, the controller saturation Octave Script was removed to better illustrate the effect.

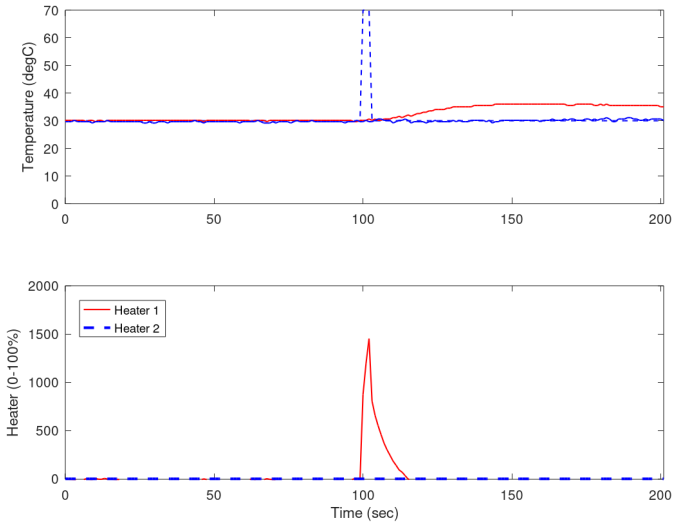

Figure 5.11: Derivative-kick TCLab test. The top plot presents both the set-point and measured temperature for transistor 1. The bottom plot presents the heater signals for transistor 1 and 2.

The maximum output the actuator reaches after the detection of the impulse input is in the region of a thousand and five hundred. Of course that in this practical example it would be impossible for the actuator to correspond to that value since it has output limitations in place. Regardless, this effect could prove more harmful in actuators that lack that kind of protection.

In Figure 5.12 the proportional derivative controller is represented in control block form. This version is a modified proportional controller simulation scheme where the values for the control parameters are changed to include the derivative time and gain constants and two control blocks are added to calculate the derivative contribution resulting from the multiplication of the continuous signal derivative by the gain constant.

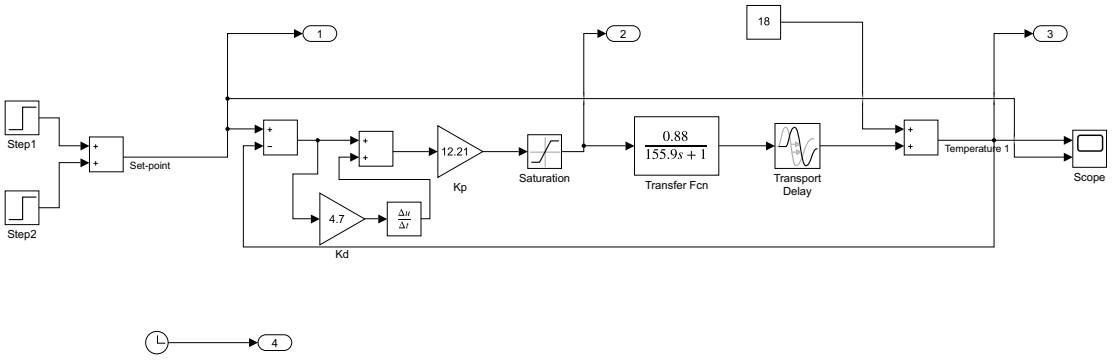

Figure 5.12: Simulink© Model for Proportional Derivative Control. Two steps are applied to the system: step 1 at t=0s with amplitude 25 degrees and step 2 at t=100s with amplitude 60 degrees.

In Figure 5.13 it is shown the validation of the results obtained in the practical experiment with only minor and expected differences in oscillations that are stipulated to originate from the physical equipment's unmodeled inertia and ambient limitations.

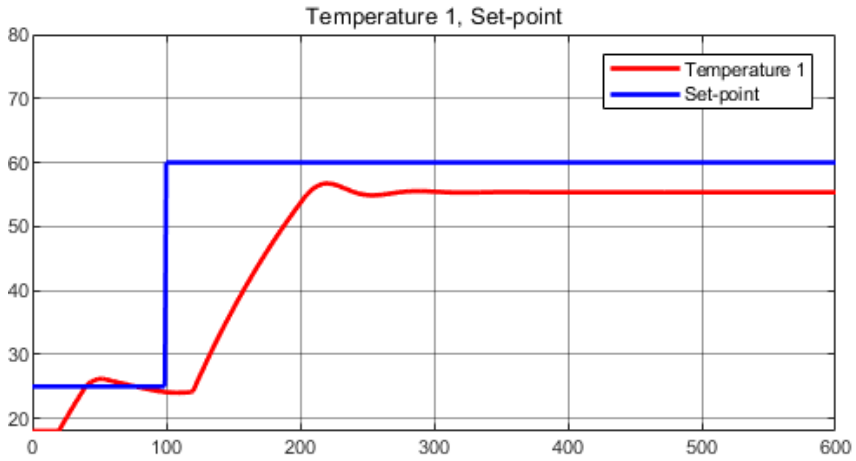

Figure 5.13: Simulated step response obtained with model presented in Figure 5.12. Proportional derivative control with Kp=12.21 and Td=4.77.

To avoid the derivative-kick effect the derivative action is applied to the controller output instead of the error signal. One important detail to note is that when applying this method the final contribution made to the controller output by the derivative controller is negative, so it subtracts from the proportional portion. This difference is due to the change from calculating the error in relation to output instead of the input. An excerpt of the Octave Script is available in Figure 5.14.

```
ek = SP_T1(i)-T1;   #Error calculation
Pk = KpD*ek;        #Proportional contribution
dyk = (T1-T1_1)/T; #Output error derivative
Dk = Kp*Td*dyk;     #Derivative contribution
Dk=(Tf/(Tf+T))*Dk_1 + (T/(Tf+T))*Dk; #Filter
Q1 = Pk-Dk;         #Output calculation

if Q1>100           #Heater saturation
  Q1=100;
end
if Q1<0
  Q1=0;
end

h1(Q1);             #Heater activation
T1_1=T1;            #Previous sample temperature reading
Dk_1=Dk;            #Previous sample derivative contribution
```

Figure 5.14: PD Anti derivative-kick Octave Script

In contrast to Figure 5.11, in Figure 5.15 the sudden jump in the set-point value doesn't translate into an excessive spike in the actuator percentage. This fact proves that the method works and the controller will not saturate or produce over bounds values in this situation.

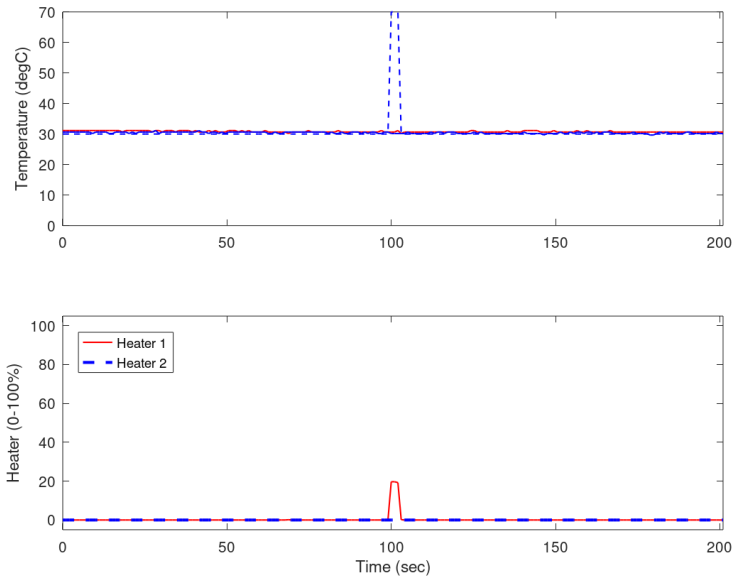

Figure 5.15: Anti derivative-kick TCLab test.The top plot presents both the set-point and measured temperature for transistor 1. The bottom plot presents the heater signals for transistor 1 and 2.

The only difference in the scheme presented in Figure 5.16 and the one in Figure 5.12 is that the derivatives contribution calculation, gain times derivative, is connected directly to the output of the control system instead of the input. With that notable exception the two schemes would be the same.

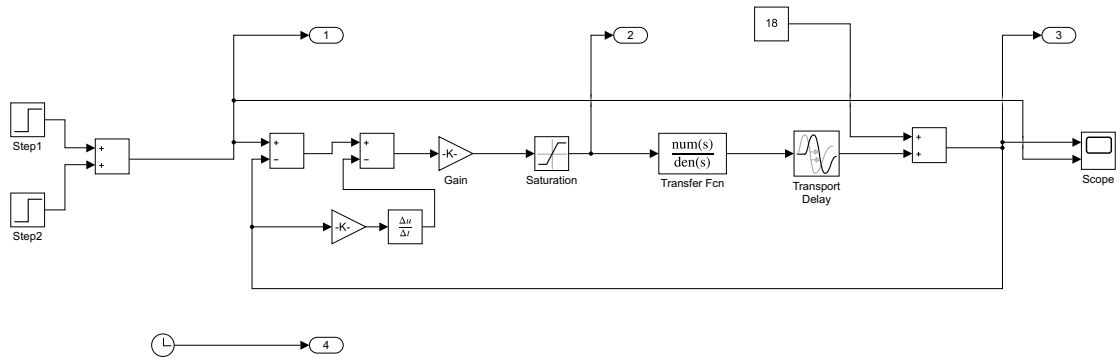

Figure 5.16: Simulink© Model for Proportional Derivative Control with anti derivative-kick. Two steps are applied to the system: step 1 at t=0s with amplitude 25 degrees and step 2 at t=100s with amplitude 60 degrees.

Finally, in the derivative control chapter, Figure 5.17 reiterates the previous observation that both noise and anti derivative methods have a very noticeable effect on the actuators fluctuations and maximum values but have little or no effect on the actual readings obtained in the temperature graphic both in a practical and simulation context.

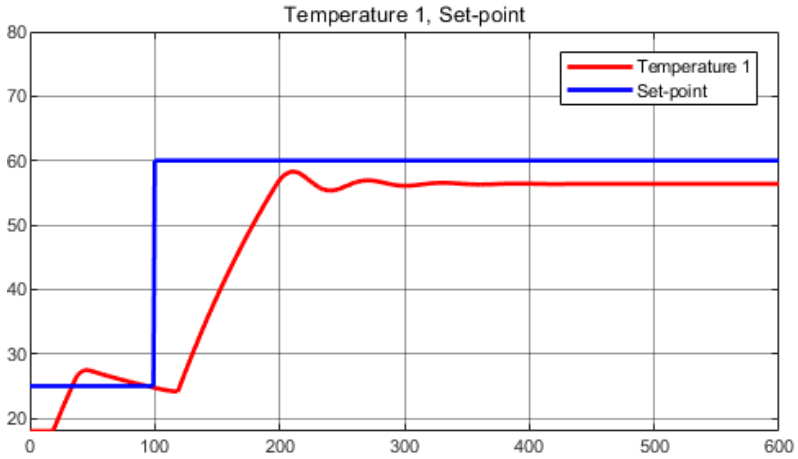

Figure 5.17: Simulated step response obtained with model presented in Figure 5.16. Proportional derivative control with anti derivative-kick and Kp=12.21 and Td=4.77.

5.4 Proportional and Integrative Control

5.4.1 Windup and Anti-Windup

In Figure 5.18 the application of the integrative component of the controller in Octave is displayed. In this Octave Script the integral contribution is calculated in the Ik factor (integrative contribution) which makes use of three different parameters, Kp (proportional gain), Ti (integrative time constant), ek (current set-point error) and zk (integer of the error). The Kp and Ti parameters are calculated based on the tuning rules of choice, in this case Cohen-Coon, the zk parameter is calculated based on the sampling time and the calculated error. This application of the PI controller doesn't possess any anti-windup method and only has actuator limitations implemented.

```
ek = SP_Tl(i)-Tl; #Error calculation
Pk = Kp*ek;       #Proportional contribution
zk =zk_1+T*ek;    #Error integer
Ik = (Kp/Ti)*zk;  #Integrative contribution
Ql = Pk+Ik;       #Output calculation

if Ql>100         #Heater saturation
   Ql=100;
 end
 if Ql<0
    Ql=0;
 end

 hl(Ql);          #Heater activation
 zk_1=zk;         #Previous error integer
```

Figure 5.18: Proportional Integrative Octave Script: Case without overshoot.

In Figure 5.19, the overshoot is massive and the settling time is very high due to an effect known as windup that occurs on controllers with an integrative portion. This effect is caused by an accumulation in the error stored by the integral when the controller doesn't take into account the actuator's physical limitations and thus expects the error to have been corrected after sending signals to be actuated beyond what is possible. This also leads to the controller being slow to react to changes in the environment. In Figure 5.19 the saturation is visible from the one hundred second mark to around two two hundred and fifty second mark where the value is limited at one hundred percent for about one hundred and fifty seconds and then suddenly drops to zero.

Table 5.5: Proportional gain and Integrative time constant values

Kp	Ti(s)
18.36	48.20

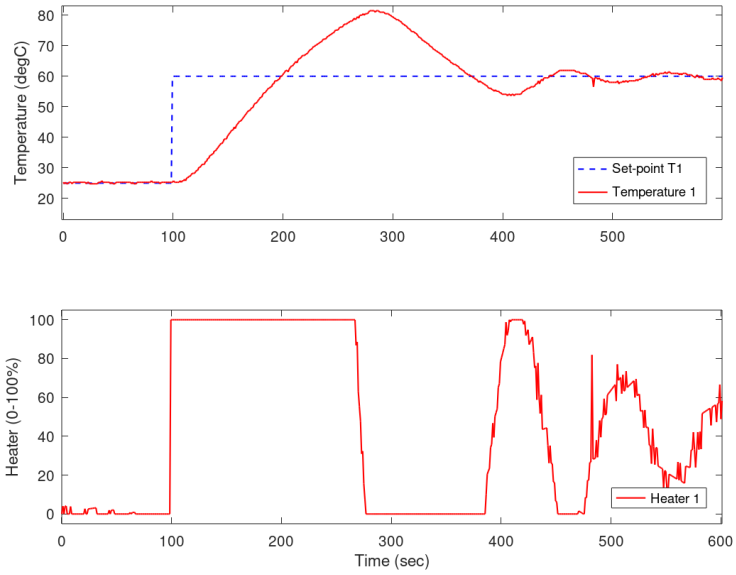

Figure 5.19: TCLab proportional integrative control test obtained with Octave: no anti-windup. The top plot presents both the set-point and measured temperature for transistor 1. The bottom plot presents the heater signals for transistor 1.

In Figure 5.20 the simulation scheme for the proportional integrative controller is displayed. This scheme was made using Simulink© and differs from the one shown in Figure 5.4 by introducing the proportional gain and integrative block multiplied by each other connected to a sum block that adds the integrative contribution to the proportional one. Also, a load disturbance block was added to test the resilience of the various controllers to this phenomenon.

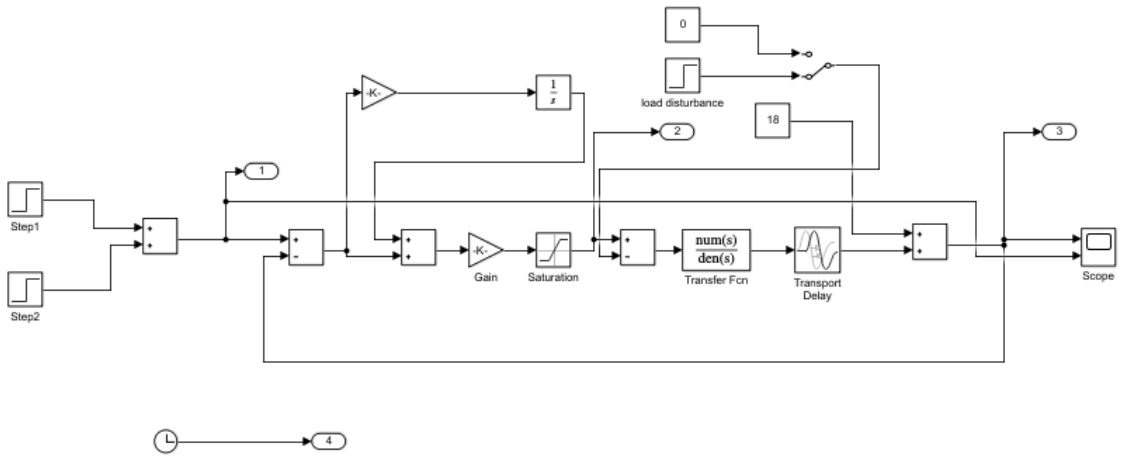

Figure 5.20: Simulink© Model for Proportional Integrative Control. Two steps are applied to the system: step 1 at t=0s with amplitude 25 degrees and step 2 at t=100s with amplitude 60 degrees.

In Figure 5.21 the confirmation of the wind-up effect is visible in the overshoot present and the overly long settling time. The effect is even more visible in the simulation than in the practical experiment.

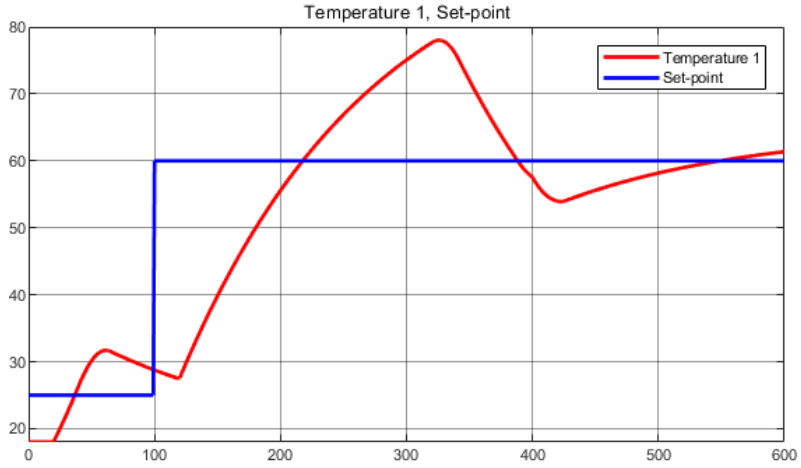

Figure 5.21: Simulated step response obtained with model presented in Figure 5.20. Proportional integrative control Kp=18.36 and Td=48.20 s.

In Figure 5.22 implements the anti-windup mechanic simply by adding a component in the saturation if clause that substitutes the value the calculated zk(integer of the error) parameter basically resetting it to the previous unsaturated value. In this manner the anti-windup effect is avoided because all the commands above the actuator's limitations are replaced by within limits commands. This method is called error reset.

```
ek = SP_T1(i)-T1; #Error calculation
Pk = Kp*ek;       #Proportional contribution
zk = zk_1+T*ek;   #Error integer
Ik = (Kp/Ti)*zk;  #Integrative contribution
Q1 = Pk+Ik;       #Output calculation

if Q1>100         #Heater saturation
   Q1=100;
   zk = zk_1;     #Error integer reset
 end
 if Q1<0
    Q1=0;
    zk = zk_1;    #Error integer reset
 end

 h1(Q1);          #Heater activation
 zk_1=zk;         #Previous error integer
```

Figure 5.22: Proportional Integrative with anti wind-up Octave Script.

In Figure 5.23 the anti-windup Octave Script is tested in a practical experiment. The result shows the elimination of the windup effect most noticeable in greatly improved overshoot and settling time. This example also demonstrates the superior advantage of the integrative controller in comparison to the previous derivative and proportional experiments in an absence of a steady-state error. The integrative controller solves that problem but lacks the ability to react as swiftly to set-point changes which is demonstrated in an inferior rising time when compared to its derivative counterpart.

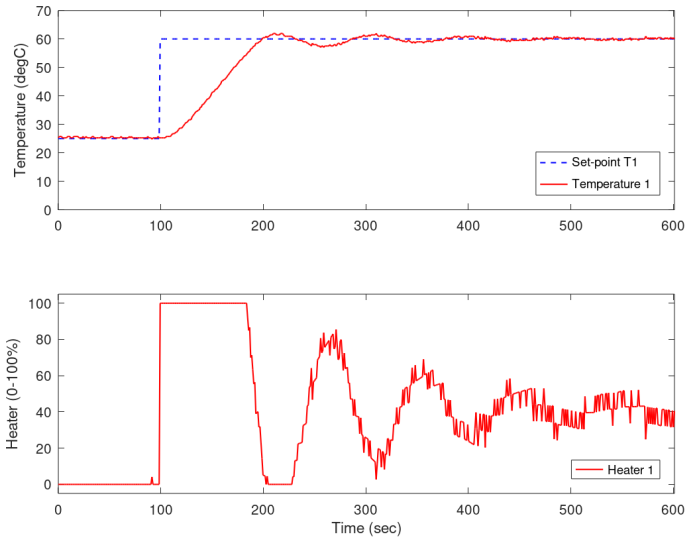

Figure 5.23: TCLab proportional integrative control test obtained with Octave:with anti-windup. The top plot presents both the set-point and measured temperature for transistor 1. The bottom plot presents the heater signals for transistor 1.

Table 5.6: Performance Indices Table: Proportional Integrative Controller

Overshoot (%)	SS Error (%)	Rising Time (s)	IAE	ITAE	TV
8.56	0.40	69.92	$3.108e+03$	$0.524e+06$	$2.996e+04$

The simulation of the anti-windup mechanic in a control block context requires additional blocks. To the blocks present in the integrative controller with no anti-windup mechanism in Figure 5.24 a kind of feedback mechanism is added in the form of an additional gain block that feeds into a sum block that is going to add the previous integrative gain to the additional one. The additional gain is previously multiplied by the product of the subtraction of the output of the saturation block by the signal that enters said saturation block.

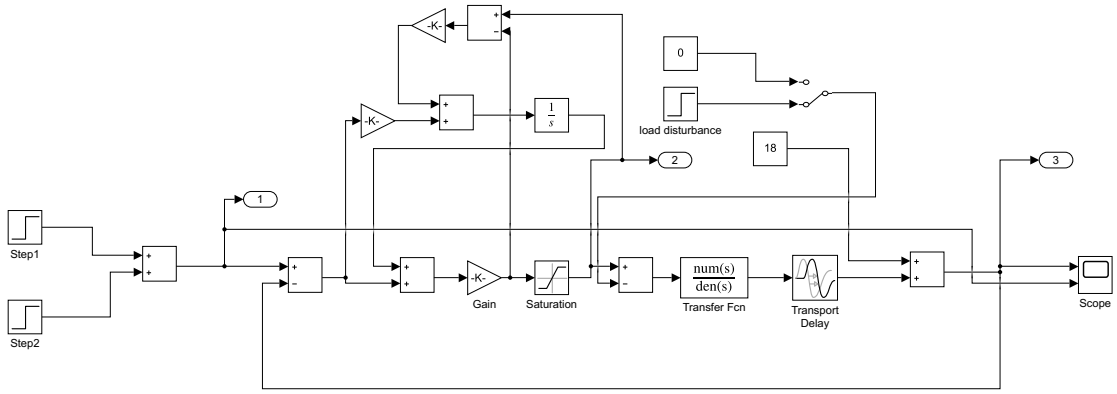

Figure 5.24: Simulink© Model for Proportional Integrative Control with anti-windup. Two steps are applied to the system: step 1 at t=0s with amplitude 25 degrees and step 2 at t=100s with amplitude 60 degrees.

In Figure 5.25, the result of the first set-point is exaggerated in the form of the overshoot present due to the absence of the hardware limitations being modelled. The lack of an overshoot compared to the practical experiment in Figure 5.23 is explained by the different gain value present when simulating other tuning methods. Regardless, the set-point following in steady-state shows no error as expected.

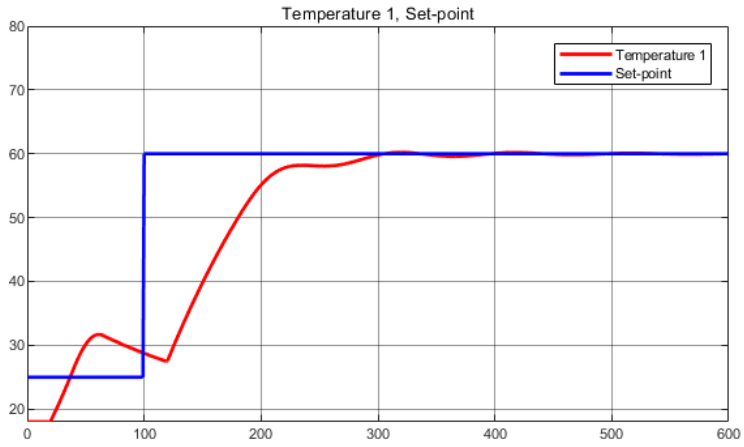

Figure 5.25: Simulated step response obtained with model presented in Figure 5.24. Proportional integrative control with anti-windup and Kp=18.36 and Td=48.20 s.

5.5 Proportional, Integrative and Derivative Control

Proportional integrative and derivative controllers (PID) join together all the advantages of the previous individual controllers. The proportional contribution is the foundation of the controller response and adjusts the ratio between the error present in the actual temperature when compared to the desired temperature and the amount of activation that should be ordered to the heaters. The derivative part increases the proportional controller's slow reaction and rise times by making the actuator react to the variation in the temperature curve instead of just the pure error magnitude. This added complexity also reduces oscillations and overshoot. The final problem that has to be solved is the fact that without the integrative portion both the proportional and derivative action in the controller possess a steady-state error. With the integrative contribution the error is stored not as a simple difference of values but as an the integral that represents the area above the temperature curve and below the horizontal line that represents the desired final temperature. This method gives the controller the ability to detect exactly the remaining amount of correction needed to reach the set-point, provided that there is an anti-windup method in place.

In Figure 5.26, an example of a PID controller Octave Script is shown. In this example both anti derivative-kick and anti-windup methods are present in the derivative and integrative parts of the controller respectively.

```
ek = SP_T1(i)-T1; #Error calculation
Pk = Kp*ek;       #Proportional contribution
zk =zk_1+T*ek;    #Error integer
Ik = (Kp/Ti)*zk;  #Integrative contribution
Q1 = Pk+Ik;       #Output calculation

if Q1>100         #Heater saturation
   Q1=100;
  end
 if Q1<0
    Q1=0;
 end

 h1(Q1);          #Heater activation
 zk_1=zk;         #Previous error integer
```

Figure 5.26: PID controller Octave Script

In Figure 5.27 the steady-state error is eliminated, the overshoot is reduced and at the same time the rising time is small. The controller presents fast quality results as expected.

Table 5.7: Proportional gain, Integrative and Derivative time constant values

Kp	Ti(s)	Td(s)
13.10	43.36	6.58

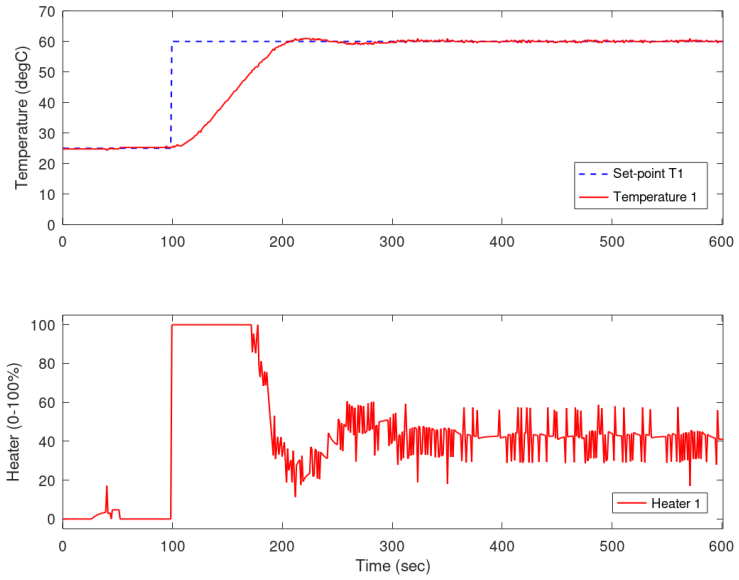

Figure 5.27: TCLab proportional integrative derivative control test obtained with Octave.The top plot presents both the set-point and measured temperature for transistor 1. The bottom plot presents the heater signals for transistor 1

Table 5.8: Performance Indices Table: Proportional Integrative Derivative Controller

Overshoot (%)	SS Error (%)	Rising Time (s)	IAE	ITAE	TV
1.56	0.47	63.74	$2.753e+03$	$0.112e+06$	$3.319e+04$

In Figure 5.28 all the previous blocks that composed the proportional, derivative and integrative schemes are joined together to form the PID controller scheme in Simulink©. The values of the gain, delay and time constant parameters are update in accordance to the tuning rules being used. Note that in this figure, the load disturbance is activated in the on position but both in the previous and next figures there was no load disturbance.

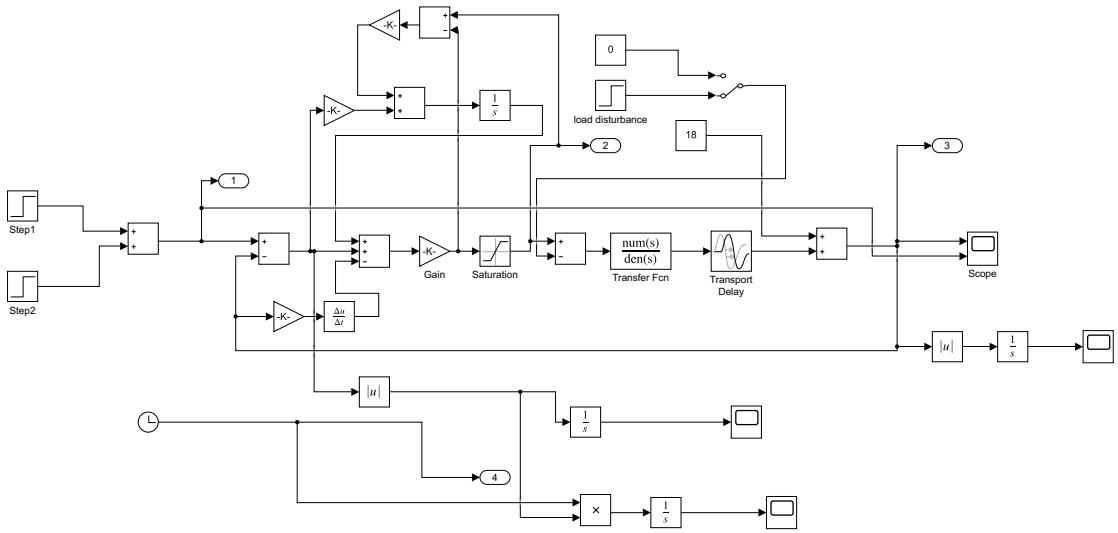

Figure 5.28: Simulink© Model for Proportional Integrative Derivative Control with load disturbance and performance metrics measurement. Two steps are applied to the system: step 1 at t=0s with amplitude 25 degrees and step 2 at t=100s with amplitude 60 degrees.And the load disturbance is applied at t=380s with amplitude -40%.

In Figure 5.29 the result is close to the experimental graphic obtained with the exception of the very noticeable overshoot in the first step attributed to a significant difference between the initial ambient temperature set in the simulation of 18 degrees and the real ambient temperature on the experiment itself that was closer to 25 degrees. Still, even with this detail taken in consideration the overshoot in the first step tends to be greater in the simulation figures when compared to the practical ones.

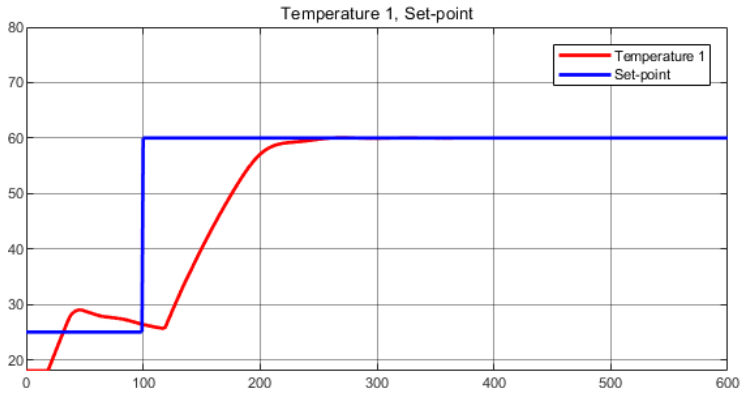

Figure 5.29: Simulated step response obtained with model presented in Figure 5.28. Proportional integrative derivative control with $Kp = 13.10$, $Ti = 43.36s$ and $Td = 6.58s$.

In the Octave Script represented In Figure 5.30 a segment is added to introduce a simulated load disturbance in the practical experiment. The objective of this change being to visualise how effectively the controller reacts to this disturbance in terms of recovery time and total integral of the error below the set-point accumulated during the recovery.

```
ek = SP_T1(i)-T1;    #Error calculation
Pk = (Kp)*ek;        #Proportional contribution
zk = zk_1+T*ek;      #Error integer
Ik = (Kp/Ti)*zk;     #Integrative contribution
dyk = (T1-T1_1)/T;   #Error derivative
Dk = (Kp)*Td*dyk;    #Derivative contribution
Dk=(Tf/(Tf+T))*Dk_1 + (T/(Tf+T))*Dk; #Filter
Q1 = Pk+Ik-Dk;       #Output calculation

if Q1>100            #Heater saturation
  Q1=100;
  zk = zk_1;         #Error integer reset
end
if Q1<0
  Q1=0;
  zk = zk_1;         #Error integer reset
end

if i>=400            #Load disturbance
  Q1=Q1-40;
end

h1(Q1);              #Heater activation
ek_1=ek;             #Previous sample error
zk_1=zk;             #Previous error integer
T1_1=T1;             #Previous sample temperature reading
Dk_1=Dk;             #Previous sample derivative contribution
```

Figure 5.30: PID Control Octave Script. A load disturbance is introduced at $t = 400$s.

In Figure 5.31 the practical experiment with a step disturbance with amplitude -40 applied in the instant four hundred seconds after the beginning of the experiment. This disturbance can be better seen in the heater graphic when a sudden drop occurs in the actuator's value, this drop is precisely 40% as programmed in the Octave Script shown previously. The controller reacts fairly quickly and recovers fully from this disturbance in under one hundred seconds, as in second five hundred it has reached the set-point again.

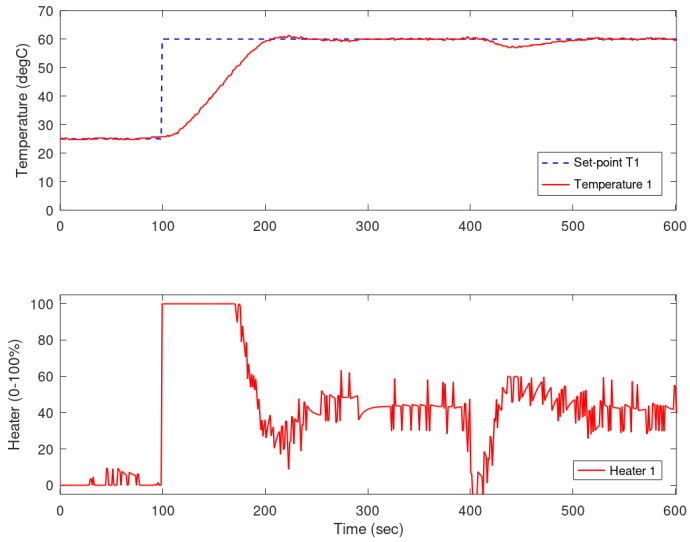

Figure 5.31: TCLab proportional integrative derivative control with load disturbance test obtained with Octave. The top plot presents both the set-point and measured temperature for transistor 1. The bottom plot presents the heater signals for transistor 1.

Table 5.9: Performance Indices Table: PID Load Disturbance Controller

IAE	ITAE	TV
$2.956e+03$	$0.205e+06$	$3.229e+04$

The simulated graphic in Figure 5.32 presents the usual discrepancy in the first step overshoot. Another more relevant difference comes in the much more pronounced depression caused by the load disturbance. In the simulation this effect is exaggerated when in contrast to the practical experiment, where the controller recovers a little more rapidly and with a slightly more symmetrical recovery curve.

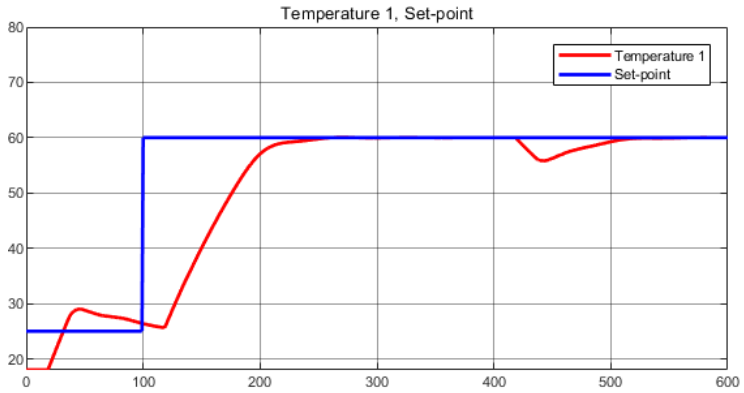

Figure 5.32: Simulated step response obtained with model presented in Figure 5.28. Proportional integrative derivative control with load disturbance. Kp= 13.10, $Ti = 43.36s$ and $Td = 6.58$.

In Figure 5.33 the first of various different tuning methods that are experimented, the AMIGO tuning method. In this case the method is focused on a smaller gain and slower rising time with a greater overshoot. The reaction in less prone to oscillations but is slower to react to and recover from the disturbance.

Table 5.10: Proportional gain, Integrative and Derivative time constants values obtained AMIGO

Kp	Ti(s)	Td(s)
5.04	65.21	9.78

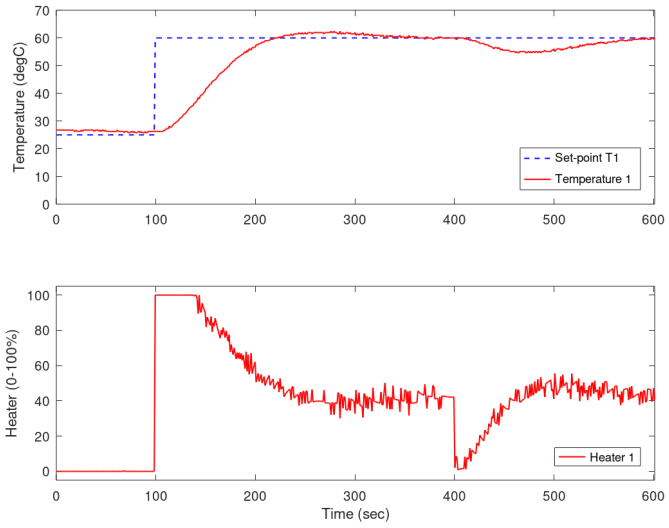

Figure 5.33: TCLab proportional integrative derivative control AMIGO tuning test obtained with Octave.The top plot presents both the set-point and measured temperature for transistor 1. The bottom plot presents the heater signals for transistor 1.

Table 5.11: Performance Indices Table: PID control with Load Disturbance. AMIGO settings.

IAE	ITAE	TV
$3.614e+03$	$0.470e+06$	$3.273e+04$

Figure 5.34 presents the result of the AMIGO tuning simulated graphic. The results are consistent with the practical experiment, a slow rise with a bigger overshoot and a long two hundred seconds recovery time. Again, the depression on the first fifty seconds after the disturbance is activated is more asymmetric than the practical experiment.

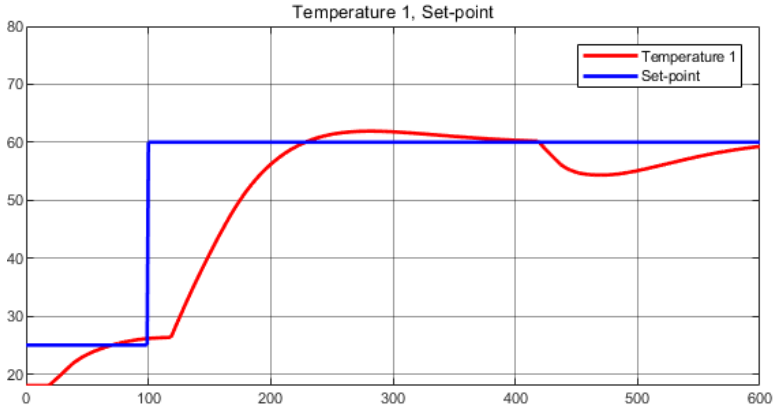

Figure 5.34: Simulated step response obtained with model presented in Figure 5.28. Proportional integrative derivative control with AMIGO tuning.$Kp = 5.04$, $Ti = 65.21$ and $Td = 9.78s$.

Figure 5.35 is the graphic of the IAE tuning method practical experiment. This graphic is the most similar to the Cohen-Coon default tuning method used in the first PID experiment and the next method ITAE also presents minor differences to the later. Compared to the AMIGO method, the rising time is significantly smaller and the recovery time is also reduced in a noticeable way. However, there is an increased in overshoot that may be attributed to the larger gain and similar time constant and delay in comparison with the Cohen-Coon method used in the reference test.

Table 5.12: PID Controller gains obtained with IAE tuning rules

Kp	Ti(s)	Td(s)
12.51	32.44	7.97

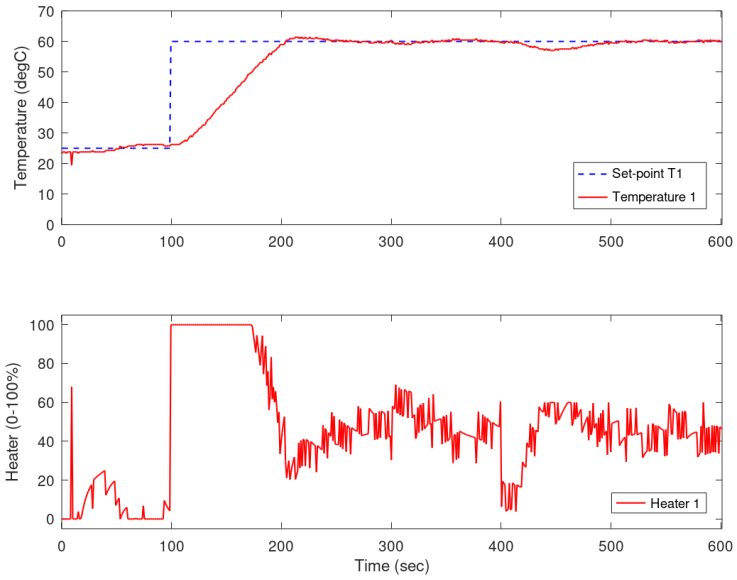

Figure 5.35: TCLab proportional integrative derivative control IAE tuning test obtained with Octave.The top plot presents both the set-point and measured temperature for transistor 1. The bottom plot presents the heater signals for transistor 1.

Table 5.13: Proportional gain, Integrative and Derivative time constant values for IAE tuning

IAE	ITAE	TV
$2.933e+03$	$0.199e+06$	$3.308e+04$

Again, Figure 5.36 mostly confirms the findings of the practical experiment. It is important to note that all of the simulations related to the different tuning methods are conducted using the same control scheme used on Figure 5.28 but with the adjusted parameters gain, time constant and delay on the respective control blocks.

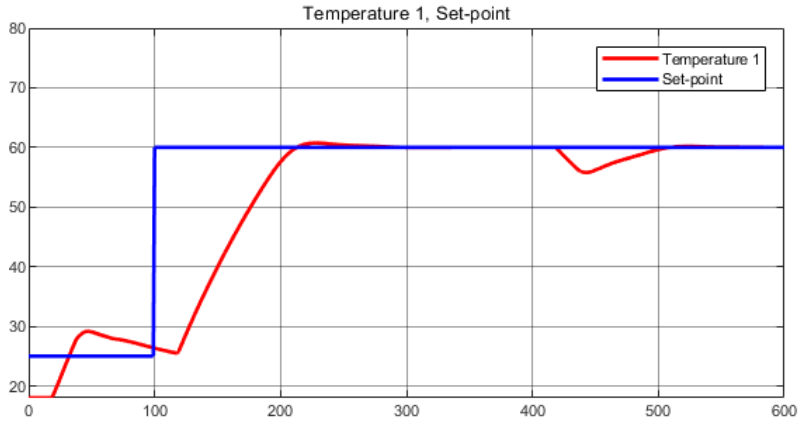

Figure 5.36: Simulated step response obtained with model presented in Figure 5.28. Proportional integrative derivative control with IAE tuning: $Kp = 12.51$, $Ti = 32.44s$ and $Td = 7.97s$.

The ITAE method has similar results to the IAE with a bigger overshoot than the one present on the reference Cohen-Coon tuning graphic and smaller than the result of the AMIGO method. The recovery time is satisfactory and the drop from the set-point value is even smaller then than both the reference and the IAE method by a slight margin and when compared to the AMIGO experiment the improvement is very noticeable.

Table 5.14: Parameter settings Table: PID ITAE tuning Controller

Kp	Ti(s)	Td(s)
12.32	34.07	8.11

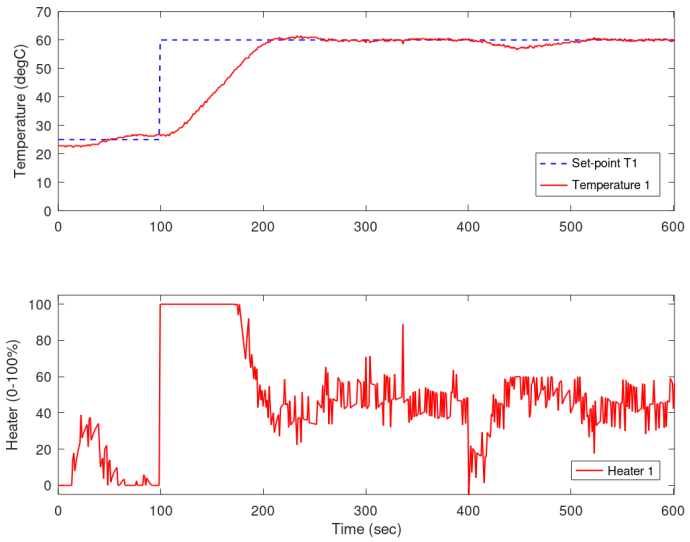

Figure 5.37: TCLab proportional integrative derivative control ITAE tuning test obtained with Octave.The top plot presents both the set-point and measured temperature for transistor 1. The bottom plot presents the heater signals for transistor 1.

Table 5.15: Proportional gain, Integrative and Derivative time constant values for ITAE tuning

IAE	ITAE	TV
$2.935e+03$	$0.201e+06$	$3.307e+04$

The ITAE simulation graphic on Figure 5.38 is a close representation of the ITAE tuning practical experiment that serves the purpose of validating the data obtained in that practical experiment. In the simulation the value of the time sample is set to one second to correspond with the value used in the Octave Script.

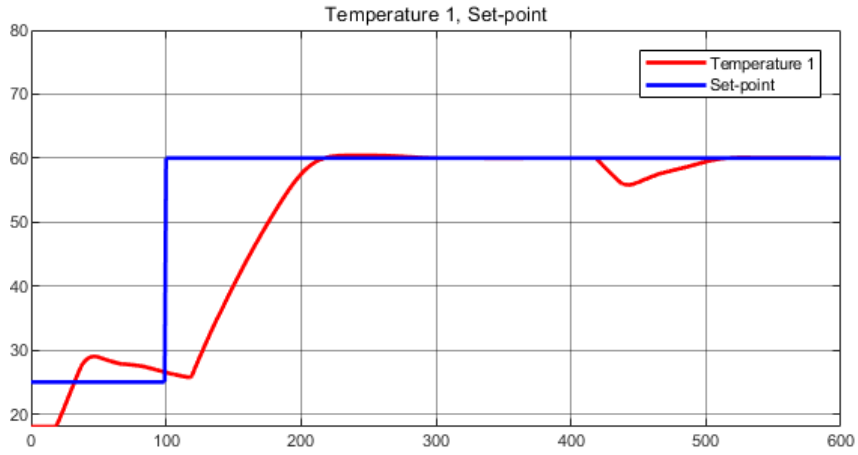

Figure 5.38: Simulated step response obtained with model presented in Figure 5.28. Proportional integrative derivative control with ITAE tuning: $Kp = 12.32$, $Ti = 34.07s$ and $Td = 8.11s$.

The final tuning method studied is SIMC. This method is more focused in creating a robust controller that despite not being the best in optimisation for particular scenarios such as the TCLab temperature control experiment it compensates that fault by being adaptable enough to be applied to other control experiments with different parameters (to a degree) and still produce satisfactory results without becoming to unstable. This makes it so that the time until the controller reaches the desired set-point temperature is delayed in relation to other tuning methods despite the rising time being close to those. The load disturbance recovery is also even slower than the one on the AMIGO tuning method that had been the longest so far. Also, the minimum point reached during the temperature drop caused by the disturbance is the lowest of the methods presented. As an advantage of being slower the SIMC tuning never produces a significant overshoot upon reaching the set-point temperature. The graphic with the experimental result is shown in Figure 5.39.

Table 5.16: PID Controller gains obtained with PID SIMC tuning rules.

Kp	Ti(s)	Td(s)
7.21	143.22	6.06

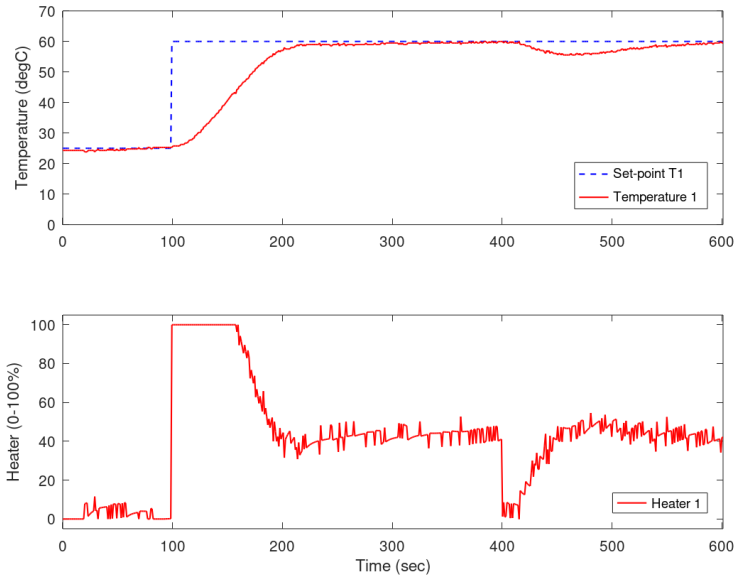

Figure 5.39: TCLab proportional integrative derivative control SIMC tuning test obtained with Octave.The top plot presents both the set-point and measured temperature for transistor 1. The bottom plot presents the heater signals for transistor 1.

Table 5.17: Proportional gain, Integrative and Derivative time constant values for SIMC tuning

IAE	ITAE	TV
$4.833e+03$	$0.682e+06$	$3.111e+04$

Figure 5.40 doesn't show big discrepancies with the results of the experiment. The gain applied is sufficient for the temperature to reach the set-point before the load disturbance activation, despite being slower. All in all, the dynamic of the controller seems to be coherent with the results produced except for the aforementioned slightly reduced gain.

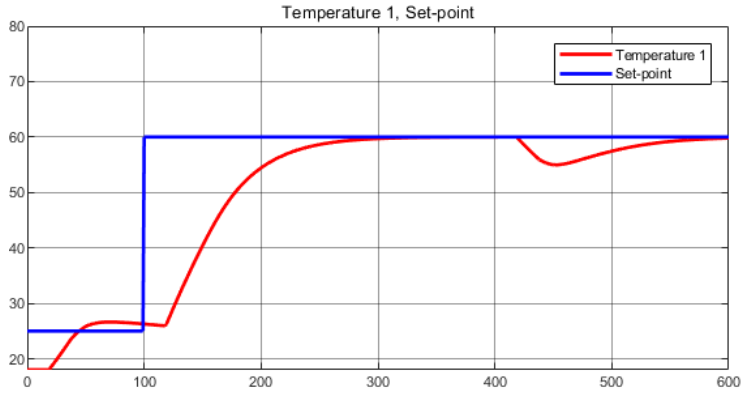

Figure 5.40: Simulated step response obtained with model presented in Figure 5.28. Proportional integrative derivative control with SIMC tuning.Kp= 7.21, Ti=65.81 and Td=6.06.

In table 5.18 the different tuning methods used in the PID controller experimental test are compared based on three different performance indices. To establish the comparison, the lower the values are for all the parameters IAE, ITAE and TV the better the controller performed. The total variation (TV) parameter represents the degree of oscillation and overall consistency of the output temperature curve. The integral of the error (IAE) represents the intersection area between the set-point line and the temperature curve. Finally, the integral of the product of the error and time variables (ITAE) represents the weighted sum of the total area of the error. What this means is that in this parameter, the later the error occurs in the experiment the more it contributes to the overall total error count and the error in the initial moments of the experiment as a much smaller influence in the final result. When analysing the results SIMC tuning obtains one of the best results in total variation, this can be attributed to a bigger focus on a robust controller with smaller reaction times and oscillations, although it performs worst than ITAE an IAE it outperforms AMIGO in the remaining parameters for accumulating the least error. The IAE and ITAE methods both come close in all parameters because the key tuning values are similar and the weighted error balances out the bigger contribution and small area of the load disturbance and the larger area and smaller contribution of the initial transitory phase. The AMIGO method has the worst error performance of all the methods and only beats IAE and ITAE in total variation. However, these tuning rules were proposed based on robustness measures. It can be considered the worst performing method both in set-point tracking and consistency. The reference Cohen-Coon base method performs worse but close to the IAE and ITAE best error performance while gaining more stability in the form of a better total variation result. This concludes that for this application in particular the Cohen-Coon tuning offers the best balance between set-point tracking performance and total variation of all methods.

Table 5.18: Performance Indices Table: PID Load Disturbance Tuning Methods

Method	IAE	ITAE	TV
Cohen-Coon	$2.956e+03$	$0.205e+06$	**3.229e+04**
AMIGO	$3.614e+03$	$0.470e+06$	$3.273e+04$
IAE	$2.933e+03$	$0.199e+06$	$3.308e+04$
ITAE	**2.935e+03**	**0.201e+06**	$3.307e+04$
SIMC	$3.359e+03$	$0.329e+06$	$3.258e+04$

Chapter 6

Conclusion and Future Work

This work addressed Portable Laboratory kits for control engineering control education teaching/learning. It was shown that small sized (pocket-sized) laboratories which can be used anywhere and anytime can greatly facilitate control engineering teaching. Within the reviewed control kits the Temperature Control Laboratory (TCLab) was the one selected for deeper study and to perform industrial control experiments. The TCLab is a resource for controller development and model identification while being a pocket-sized lab with the purpose of reinforcing control theory for students. Many universities around the world have adopted this lab for process control education. The lab is also used for the online courses Process Dynamics and Control and Dynamic Optimisation. Putting in practice the learned concepts of process control using the TCLab is relatively simple since this kit has many support from previous researchers as well as many free and open-source platforms to preform control experiences. All platforms used in this project are very user-friendly as well as beginner-friendly making this a very powerful kit when teaching students the development and the programming of control algorithms applied to various devices. Since students have different levels of technical knowledge (both in hardware or software), using Arduino-based practices, it is possible to affirm that students can better learn lacking skills. Thus, students who have higher level of knowledge in hardware can learn object-oriented programming while those who have higher level on programming can enhance their knowledge on the hardware. In addition, being the Arduino hardware low cost and having a free-to-use software, encourages students to purchase their own devices in order to practice outside the classroom environment as well as to develop personal projects and, consequently, develop their personal technical skills. As the Arduino is an exceptional educational tool, it is not desirable to completely leave aside the laboratory equipment, since it is likely that students will contact in their professional career. The objectives proposed for this project were successfully reached, with relevance given to the following:

- Performing a review of Arduino based control kits;
- TCLab study, programming and testing in several environments, namely: Arduino©, Matlab©, Simulink©, Gnu OCTAVE and Python;

- Industrial controller test using On-Off, P, PI, PD and PID control;
- A comparison of several tuning methods using TCLab;

The hardest difficulties encountered, while executing this project, were to recreate in a accurate way all the control experiments while demonstrating their strengths and downsides in a clear manner and explain how to setup those experiments and to make the kit work in all the different open-source and licensed software.

In the future, the main work to be done is mainly focused in some control methods left of that could also be implemented, feed-forward control and cascade control. Those methods would be useful to demonstrate some techniques that utilise more complex controller loops when compared to the ones already addressed and are able to react to outside interference in a more adequate way and in a greater number. In terms of software applications, the Scilab software is also an interesting candidate to explore as it possesses different characteristics from the tools in this dissertation, increasing the overall variety while also being open-source. Regarding the requirements set in the beginning of this project, it is possible to affirm that they were fulfilled completely, concluding that the TCLab kit is indeed effective in the teaching of control experiments in a domestic environment and using a variety of different tools.

6.1 Source Code

For more information and to access the scripts used throughout this dissertation a website is available in the following link: `http://fe.up.pt/asousa/tclab`.

References

[1] B.Ll. Jones J. D. Hedengren (2019) J. A. Rossiter, S.A. Pope. *Evaluation and demonstration of take home laboratory kit*. Invited Session: Demonstration and poster session, 12th IFAC Symposium on Advances in Control Education, July 7-9, 2019. Philadelphia, PA, USA. (Accessed on 23/03/2020).

[2] Christopher G. Braun. Experiments on the cheap: Using a student data acquisition system. Colorado School of Mines. (Accessed on 28/01/2020).

[3] Carion Pelton. Take home labs and the ball and beam. Oklahoma State University, Stillwater, Oklahoma, 2012. (Accessed on 09/02/2020).

[4] Hedengren J. D. *Dynamics and Control: Arduino Setup and Installation*. https://apmonitor.com/pdc/index.php/Main/ArduinoSetup. (Accessed on 2/05/2020).

[5] Anaconda Enterprise (2020). *Anaconda Documentation: Tutorial*. https://docs.anaconda.com/anaconda/navigator/tutorials/. (Accessed on 24/02/2020).

[6] Mathworks MATLAB Hardware Team. *Arduino Support from MATLAB*. https://www.mathworks.com/hardware-support/arduino-matlab.html?s_tid=AO_HS_info. (Accessed on 7/03/2020).

[7] Hedengren J. D. (2019). *Temperature Control Lab Kit*. https://apmonitor.com/heat.htm, note = (Accessed on 15/02/2020).

[8] Christopher G. Braun. *Experiments on the Cheap: Using a Student Data Acquisition System*. Colorado School of Mines. (Accessed on 20/05/2020).

[9] P.B Moura Oliveira and J. D. Hedengren (2019). "an apmonitor temperature lab pid control experiment for undergraduate students". 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), pp. 790-797, DOI: 10.1109/ETFA.2019.8869247. (Accessed on 10/12/2019).

[10] Dynamics control: Temperature control lab. http://apmonitor.com/pdc/index.php/Main/ArduinoTemperatureControl. (Accessed on 13/11/2019).

[11] Kantor J.C. Reuel N. (2019) Hedengren J. D., Martin R. A. *Temperature Control Lab for Dynamics and Control*. AIChE Annual Meeting, Orlando, FL, Nov 2019. (Accessed on 3/03/2020).

[12] Martin R.A. Kelly J.D. Hedengren J.D. (2020) Park, J. *Benchmark Temperature Microcontroller for Process Dynamics and Control*. Computers Chemical Engineering, Special Issue in Honor of Thomas F. Edgar, 135, doi: 10.1016/j.compchemeng.2020.106736. (Accessed on 17/03/2020).

[13] (2019) Moura Oliveira P. B. e Hedengren J. D. *An APMonitor Temperature Lab PID Control Experiment for Undergraduate Students*. 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), pp. 790-797, DOI: 10.1109/ETFA.2019.8869247. (Accessed on 5/04/2020).

[14] J. D. Hedengren Moura Oliveira P. B. and J. A. Rossiter (2020). *Introducing Digital Controllers to Undergraduate Students using the TCLab Arduino Kit*.

[15] J. D. Hedengren Moura Oliveira P. B. and J. Boaventura Cunha (2020). *Bridging Theory to Practice: Feedforward and Cascade Control with TCLab Arduino Kit*. Aceppted for the Controlo-2020 14th International Conference on Automatic Control and Soft Computing, Bragança, Portugal | July 01-03, 2020. (Accessed on 22/04/2020).

[16] B.Ll. Jones J. D. Hedengren (2019) J. A. Rossiter, S.A. Pope. *Evaluation and demonstration of take home laboratory kit*. http://eprints.whiterose.ac.uk/146019/1/ace19demohedfinal.pdf. (Accessed on 13/01/2020).

[17] Hedengren J. D. *Python bindings for the BYU Arduino Temperature Control Lab*. https://pypi.org/project/tclab/?fbclid=IwAR2EFiH7pFaUsWSvNnokPJhOwHzrASnZyfQ25_UwYssCHgubI4D2RYal9tc. (Accessed on 10/02/2020).

[18] Python Software Foundation (2020). *What is Python?* https://www.python.org/doc/essays/blurb/. (Accessed on 13/05/2020).

[19] Hedengren J. D. (2019). *Python bindings for the BYU Arduino Temperature Control Lab*. https://pypi.org/project/tclab/. (Accessed on 27/12/2019).

[20] Hedengren J. D. (2018). *Arduino Temperature Control Lab for Simulink and Matlab*. https://www.mathworks.com/matlabcentral/fileexchange/65760-arduino-temperature-control-lab-for-simulink-and-matlab. (Accessed on 28/05/2020).

[21] B. Thomas (2004). *Ziegler-Nichols Method*. https://pages.mtu.edu/~tbco/cm416/zn.html. (Accessed on 16/03/2020).

[22] Payne Lee (2014). *Tuning PID control loops for fast response*. https://www.controleng.com/articles/tuning-pid-control-loops-for-fast-response/. (Accessed on 4/05/2020).

[23] P.B Moura Oliveira and J. D. Hedengren (2019). "introducing digital controllers to undergraduate students using the tclab arduino kit.". (Accepted for the IFAC-2020 World Congress). (Accessed on 21/12/2019).