A REPORT

ON

DESIGNING AND DEVELOPMENT OF RIVITING ANALYSER

BY

Vishal Srivastava   2010A3PS224G

AT

Titan Company Ltd.

HOSUR

A Practice School-II Station of

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI

December 2013

A REPORT

ON

DESIGNING AND DEVELOPMENT OF RIVITING ANALYSER

BY

Vishal Srivastava	2010A3PS224G	B.E (Hons.) Electrical and Electronics

Prepared in partial fulfilment of the

Practice School-II

Course No. BITS C412/C413

AT

Titan Company Ltd.

HOSUR

A Practice School-II Station of

BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI

December 2013

ACKNOWLEDGEMENT

I am thankful to Mr Benjamin Felix, Manager-HR Titan Company for providing this opportunity. I would like to express my sincere thanks to Mr Natrajan, Project Head, Project Division for allotting me in Machine Building Activity.

I express my sincere thanks to Mr Ravidas Bhatt, Electrical Head, Center of Excellence, for being a supportive mentor and appreciating as well as enriching my ideas with his vast experience. I should also like to thank Mr S Rajesh for providing inputs and ideas wherever required in the project.

I would like to express my sincere thanks to Mr Siju CR for his constant support and guiding me time to time

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE

PILANI (RAJASTHAN)

PRACTICE SCHOOL DIVISION

Station: Titan Company Ltd.   **Center:** Hosur

Duration: July 4th to Dec 14th 2013   **Date of Start:** July 4th

Date of submission: 06th Dec 2013

Title of the Project: Designing and Development of Riveting Analyser.

Name: Vishal Srivastava   **ID No.:** 2010A3PS224G

Discipline: B.E. (Hons.) Electrical and Electronics Engineering

Name of the Expert: Mr Ravidas Bhatt   **Designation:** Electrical Head

Name of the PS Faculty: Mr Siju CR

Keywords: Riveting Controller

Project Areas: Product Development

Abstract: The riveting machine is used to provide a permanent mechanical fastening solution through smooth cylindrical shaft with head called rivets. Our purpose is to make an intelligent standalone controller device which can control this machine as well as log data and produce graphical representation for the same. This report contains detailed description of the design of riveting controller followed by challenges involved with it. Finally a brief description on the future design plans is discussed.

Signature of the Student   Signature of the Faculty

Date: Date:

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE

PILANI (RAJASTHAN)

PRACTICE SCHOOL DIVISION

Response option Sheet

Station: Titan Industries Ltd   **Centre:** Hosur

ID No.: 2010A3PS224G   **Name:** Vishal Srivastava

Title of the Project: Designing and Development of Riveting Controller

Usefulness of the project to the on-campus courses of study in various disciplines. Project should be scrutinized keeping in view the following response options. Write Course No. and Course Name against the option under which the project comes.

Refer Bulletin for Course No. and Course Name.

Code No.	Response Options	Course No. & Name
1.	A new course can be designed out of this project	No
2.	The project can help modification of the course content of some of the existing Courses	Yes
3.	The project can be used directly in some of the existing Compulsory Discipline Courses (CDC)/ Discipline Courses Other than compulsory (DCOC)/ Emerging area (EA)/ Technical Art (TA) and Core Courses	Yes
4.	The project can be used in preparatory courses like analysis and application Oriented Courses(AAOC)/ Engineering Science (ES)/ Technical Art (TA) and Core Courses	Yes
5.	This project cannot come under any of the above mentioned options as it relates to the professional work of the host organization	No

_____   _____

Signature of Student   Signature of Faculty

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. About Titan Industries

Titan Company Limited began its journey in 1984 in watch industry as a joint venture between Tata and the Tamil Nadu Industrial Development Corporation. Later in 1995 the company diversified into jewellery under the brand Tanishq to capitalise on a fragmented market operating with no brands in urban cities. Today company has emerged as leading brand in watch industry having a market share of 60% in India and also has sold in about 40 countries through market subsidiaries.

Now company has even established itself into eyewear, bags and belts etc. under the brand name of fastrack.

Before that in 1990, to address the needs of the watch manufacturing and assembly a small assembly unit was set up. Over the years it delivered more than 200 high precision assembly machines and testing equipment for watch production. Later in 2004, this unit was established as a strategic business unit under the name of Titan automations. Today it caters to needs of assembly and testing lines of various industry segments such as Automotive, Electrical, Medical and Health Care and other engineering.

2. Introduction

Over the years Titan automation has grown from in house watch precision industry to a global automation solution industry. The company has diversified itself to provide automation solutions to industries such as Automotive, Electrical, Medical and Health Care and other engineering.

Due to this diversification the company now aims to become self-sufficient automation industry. For this development of various innovative in house project ideas have cultivated in the company. One such idea is the riveting controller that is planned to be used for controlling in house riveting machine.

2.1 Objective

Till now riveting machine has been only employed using a central logic control unit such as PLC (Programmable Logic Unit) in the assembly lines.

Figure 1: Assembly Line with Orbital Riveting Machine

Managing process along with its numerous sensors is itself complicated;

this generally leads to overshooting of processing overheads leading to time lags when employed in the assembly line. In order to reduce this processing overhead a standalone controller device was proposed which only caters to the riveting process.

Figure 2: Riveting Machine with Controller

So a Riveting controller has been proposed as a slave for central control unit or a master. This can significantly reduce the processing loads on the central control unit which can significantly reduce the lead time in the production.

2.2 Project Methodology

A brief idea of decision making process has been depicted in the below figure. It involved detailed study of riveting process. This was followed by additional requirement identification and data representation techniques. After identification a thorough study was conducted to find available solutions. After that a practical design was proposed for riveting controller. In later stage implementation of the design will take place.
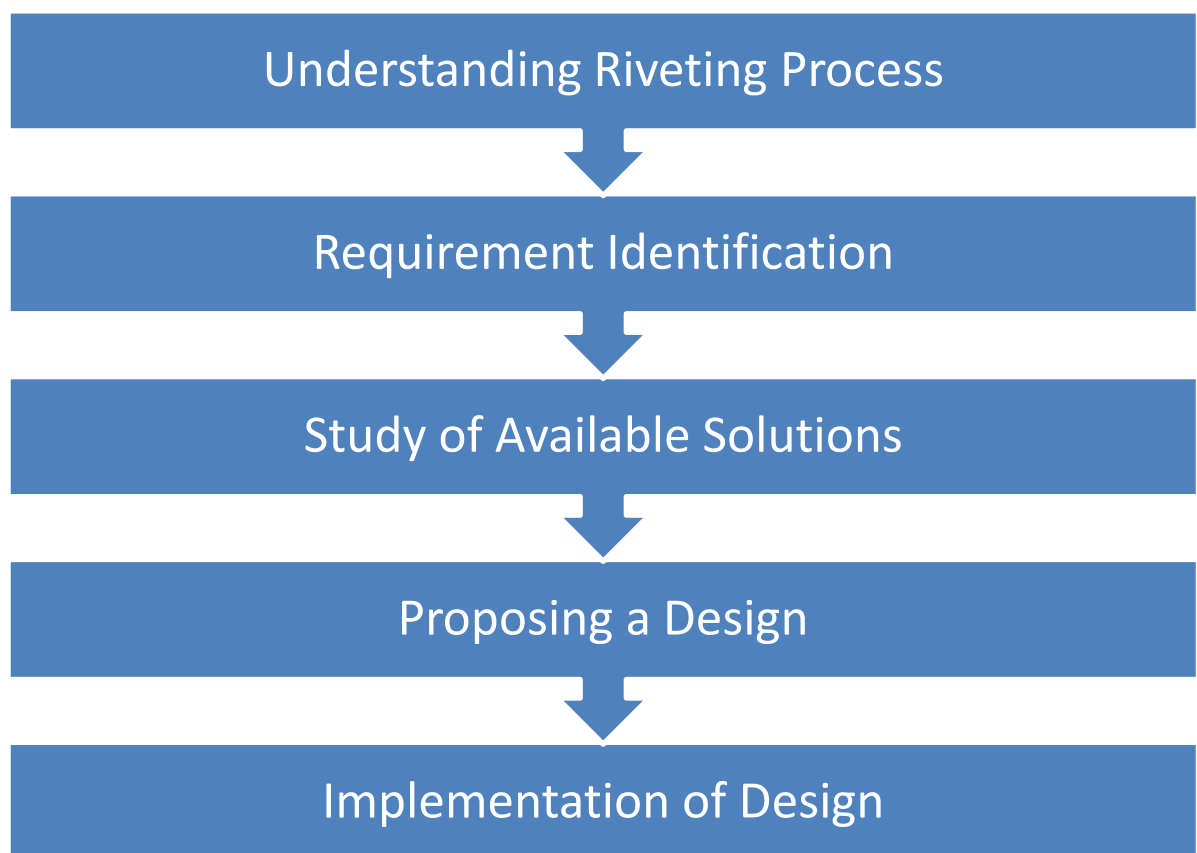
Understanding Riveting Process

Requirement Identification

Study of Available Solutions

Proposing a Design

Implementation of Design

Figure 3: Project Methodology

A detailed plan is as follows:

1. Understanding Riveting Process
   The target process was studied as to what are all the steps in the process and how is the riveting machine able to perform riveting with help of its sensor and actuators.

2. Requirement Identification
   A detail plan was asked as to what all is expected from the controller. On the basis of these requirements further decision regarding the design was to be taken

3. Study of Available Solutions
   After understanding the requirements a research was done to find for available designs for circuit and devices that can be used for its construction.

4. Proposing a Design
   A design was finally proposed based on our earlier study. This design is currently under thorough checking for its feasibility with the available resources. Then this design shall be approved after small rectifications

5. Implementation of Design
   This design shall then be implemented and fabricated.

3 Design Requirement

The device is motivated by a third party Riveting Controller fabricated by Baltec Industries. As the device is widely used in industry and was costing a fortune to the company they decided to produce one for themselves. As the device was meant as a replacement for the Baltec product, the basic configurationally requirement to be replicated was taken from it.

3.1 Processing Capabilities

The control device was supposed to handle 28 Digital as well as 3 Analog Inputs and Outputs with the minimum scan time of 10 kHz. The device should be able to hold 38 programs configurations with ability to log data within as well as an external media. The device was supposed to have a GUI (Graphical User Interface) with the capability to plot 5 input readings wrt.to time. The device was expected to have certain 8 timers with 10ms resolution and numerous counters.

From the above proposed requirements the main requirements identified were:

- 26 Digital I/Os
- 2 High speed inputs for encoder input
- 2 Analog Outputs
- Data logging
- LCD based HID
- Analysis and Graphical representation
- Porting of data files to computer through Ethernet and USB Drives

3.2 Display Unit and Graphical User Interface

For a display unit, the required device should be able to provide a wide good definition interface. As graphical representations should be clear a display of more than 5 inch was required with an inbuilt touch input. As all the control data was to be provided only by the display unit so a touchscreen was required. Also for a capable User Interface to work a touchscreen with high touch resolution was required thus a higher size was a preference.

As for Graphical User Interface, it is supposed to provide all the riveting parameters and functions within the program. These readings are supposed to be provided on the screen on a real time system. It is required to provide data logging facility and exporting to an excel sheet. Also it is supposed to monitor and display any error or warnings as alarms. A basic administration system was also to be provided to for producing new riveting program as well as manage various features for the system.

So, the main requirements that were required are:

- o Large Screen Size (more than 5 inch) with good display and touch interface.
- o Graphical User Interface that supports display of various readings as well as graphs with administration settings.

3.3 Interface

Mainly four interfaces were required

1. Raw digital input and output
   Raw digital input and outputs were required for sending control bit and status flag from Riveting controller to the PLC. These inputs were used for mode selection of the riveting controller. Also, it is to be used to interface various sensors

2. Analog input and output
   Used for interface some of the analog sensors such as LVDT.

3. Ethernet
   It is supposed to be mainly used for data logging to the PC, but can also be used for debugging purposes.

4. USB
   To be used for porting programs, graphs, readings, and alarm history from one machine to another.

3.4 Other requirements

Other requirement such as auto compensation modes has to be incorporated in order to handle deformed or undersized rivets in order to minimize wastage and increase production.

4. Available Options

4.1 Processing Unit

In terms of processing capabilities several options were available such as DSP (Digital Signal Processor), FPGA (Field Programmable Gate Array) and Microprocessors. A small briefing of the products is as follows:

1. Digital Signal Processor

Figure 4: A typical digital processing system

A Digital Signal Processor is a specialized microprocessor that has an architecture which is optimized for the fast operational needs of digital signal processing. A Digital Signal Processor (DSP) can process data in real time, making it ideal for applications that can't tolerate delays. Digital signal processors take a digital signal and process it to improve the signal into clearer sound, faster data or sharper images. Digital Signal Processors use video, voice, audio, temperature or position signals that have been digitized and mathematically manipulate them.

2. FPGA

Figure 5: Field Programable Gate Array

Field Programmable Gate Arrays (FPGAs) are semiconductor devices that are based around a matrix of configurable logic blocks connected via programmable interconnects. FPGAs can be reprogrammed to desired application or functionality requirements after manufacturing.

3. Microprocessor

Figure 6: General Purpose Processor

Microprocessors that are capable of performing a wide range of tasks are called general purpose microprocessors. General purpose microprocessors are typically the kind of CPUs found in desktop computer systems. These chips typically are capable of a wide range of tasks (integer and floating point arithmetic, external memory interface, general I/O, etc).

4.2 Display and User Interface

There are several display options available in the field of embedded display

1. LED based display

Figure 7: LED based display

An LED display is a flat panel display, which uses light-emitting diodes as a video display. An LED panel is a small display, or a component of a larger display. They are typically used outdoors in store signs and billboards, and in recent years have also become commonly used in destination signs on public transport vehicles or even as part of transparent glass area.

2. TFT Display

Figure 8: TFT Display

A thin-film transistor (TFT) is a special kind of field-effect transistor made by depositing thin films of an active semiconductor layer as well as the dielectric layer and metallic contacts over a supporting (but non-conducting) substrate.

3. LCD Display

Figure 9: LCD Display

A liquid-crystal display (LCD) is a flat panel display, electronic visual display, or video display that uses the light modulating properties of liquid crystals. Liquid crystals do not emit light directly. LCDs are available to display arbitrary images (as in a general-purpose computer display) or fixed images which can be displayed or hidden, such as preset words, digits, and 7-segment displays as in a digital clock.

4.3 Interface

Various Interfaces are available that can be industrially employed. Brief descriptions of some of these are:

1. CAN

   CAN (Controller Area Network) is a serial data bus for real-time applications. CAN was originally developed for use in the automotive industry (and is sometimes called Car Area Network) but is receiving widespread use in a wide variety of embedded applications like industrial control where high-speed communication is required.

2. RS-485

   Digital communications networks implementing the RS-485 standard can be used effectively over long distances and in electrically noisy environments. Multiple receivers may be connected to such a network in a linear, multi-drop configuration.

3. RS-232

   RS-232 is the traditional name for a series of standards for serial binary single-ended data and control signals connecting between DTE (data terminal equipment) and DCE (data circuit-terminating equipment, originally defined as data communication equipment).

4. Firewire

   The IEEE 1394 or Firewire interface is a serial bus interface standard for high-speed communications and isochronous real-time data transfer.

5. USB

   Universal Serial Bus (USB) is an industry standard developed in the mid-1990s that defines the cables, connectors and communications protocols used in a bus for connection, communication, and power supply between computers and electronic devices.

6. Ethernet

   Ethernet is a family of computer networking technologies for local area networks (LANs). Ethernet was commercially introduced in 1980 and standardized in 1985 as IEEE 802.3. Ethernet has largely replaced

competing wired LAN technologies.

4.4 Software

For embedded programming various programming language and Software are now available. Some of them are:

1. MATLAB
   MATLAB (matrix laboratory) is a numerical computing environment and fourth-generation programming language. Developed by MathWorks, MATLAB allows matrix manipulations, plotting of functions and data, implementation of algorithms, creation of user interfaces, and interfacing with programs written in other languages, including C, C++, Java, and Fortran.

2. Simulink
   Simulink, developed by MathWorks, is a data flow graphical programming language tool for modeling, simulating and analyzing multidomain dynamic systems. Its primary interface is a graphical block diagramming tool and a customizable set of block libraries.

3. Embedded C
   Embedded C is a set of language extensions for the C Programming language by the C Standards committee to address commonality issues that exist between C extensions for different embedded systems.

4. Assembly Language
   An assembly language is a low-level programming language for a computer, or other programmable device, in which there is a very strong (generally one-to-one) correspondence between the language and the architecture's machine code instructions.

5. Operating Systems
   An operating system (OS) is a collection of software that manages computer hardware resources and provides common services for computer programs. The operating system is an essential component of the system software in a computer system. Application programs usually require an operating system to function.

5. Final Decision

5.1 Processing Unit

In order to meet the above requirement certain types of processing units were researched. A comparison was made between DSP (Digital Signal Processor), FPGA (Field Programmable Gate Array) and Microprocessors on the basis of these parameters.

1. Internal Memory
2. Robustness
3. Interfacing Options
4. Mathematical computational Ability
5. Logic Computational Ability
6. Logic Implementation
7. Software
8. Processing
9. Cost

Here is the table:

Table 1: Processor Comparison

	DSP	**Microprocessor**	**FPGA**
General Use	Designed for special purpose	Used for generic usage	Used for chip designing
Internal memory	Don't have on chip flash	Have on chip flash	Don't have on chip flash
Robustness	Not robust	Highly Robust	Not robust
Interfacing Options	Limited	Several	Limited
Mathematical computational ability	Very High	Medium	High
Logic computational ability	High	Medium	Very High
Logic Implementation	Hardwired and Software	Software	Hardwired
Software	Proprietary	Open source	Proprietary
Processing	Sequential	Sequential	Parallel
Cost	High	Low	High

As we can see that there is a small difference between them. But as for our application which is neither logic nor mathematically intense, it is better to go with microprocessor implementation.

Also low cost, interfacing options and memory acted as our selection criteria for the above decision.

After this various Microprocessors were studied for implementation. For this processors from various companies were researched and finally we came to a conclusion that processors from Texas Instruments were the most appropriate for

our implementation. For selection of processors from Texas Instruments following parameters were taken into considerations:

1. Maximum clock speed
2. Graphics Accelerators
3. Cache Size
4. On chip Memory
5. General Purpose Memory
6. Supported Display Options
7. On chip Timers
8. Peripheral Supports such as USB, I2C, SPI, Ethernet etc

Here is a detailed comparison between different Texas Instrument SOCs in the market:

Table 2: Microprocessor Comparison

SOC	AM3505 - Sitara ARM Microprocessor (MPU)[3]	AM3359 - Sitara ARM Cortex-A8 Microprocessor[4]	AM3358 - Sitara ARM Cortex-A8 Microprocessor[4]	AM3357 - Sitara ARM Cortex-A8 Microprocessor[4]	AM3356 - Sitara ARM Cortex-A8 Microprocessor[4]
Status	ACTIVE	ACTIVE	ACTIVE	ACTIVE	ACTIVE
SubFamily	AM35x ARM Cortex-A8	AM335x ARM Cortex-A8	AM335x ARM Cortex-A8	AM335x ARM Cortex-A8	AM335x ARM Cortex-A8
Applications	Consumer Electronics, Industrial, Medical	Connected Vending Machines, Home/Building Automation, Consumer Electronics	Portable Navigation, Connected Vending Machines, Home/Building Automation, Consumer Electronics	Portable Navigation, Connected Vending Machines, Home/Building Automation, Consumer Electronics	Connected Vending Machines, Home/Building Automation, Consumer Electronics
Operating Systems	Neutrino, Integrity,Windows Embedded CE, Linux, VXWorks, Android	Linux, Android,Windows Embedded CE	Linux, Android,Windows Embedded CE	Linux, Android, Windows Embedded CE	Linux, Android, Windows Embedded CE
ARM CPU	1 ARM Cortex-A8	1 ARM Cortex-A8	1 ARM Cortex-A8	1 ARM Cortex-A8	1 ARM Cortex-A8
ARM MHz (Max.)	600	800	600, 800, 1000	300, 600, 800	300, 600, 800
ARM MIPS (Max.)	1200	1600	1200, 1600, 2000	600, 1200, 1600	600, 1200, 1600
Graphics Acceleration		1 3D	1 3D		
Crypto HW Accelerators Available	Yes				
On-Chip L1 Cache	32 KB (ARM Cortex-A8)	64 KB (ARM Cortex-A8)	64 KB (ARM Cortex-A8)	64 KB (ARM Cortex-A8)	64 KB (ARM Cortex-A8)
On-Chip L2 Cache	256 KB (ARM Cortex-A8)	256 KB (ARM Cortex-A8)	256 KB (ARM Cortex-A8)	256 KB (ARM Cortex-A8)	256 KB (ARM Cortex-A8)
Other On-Chip Memory	64 KB	128 KB	128 KB	128 KB	128 KB
Display Options		LCD	LCD	LCD	LCD
General Purpose Memory	1 16-bit GPMC, 1 32-bit (SDRC, Async SRAM, NAND flash, NOR flash, OneNAND flash)	1 16-bit (GPMC, NAND flash, NOR Flash, SRAM)	1 16-bit (GPMC, NAND flash, NOR Flash, SRAM)	1 16-bit (GPMC, NAND flash, NOR Flash, SRAM)	1 16-bit (GPMC, NAND flash, NOR Flash, SRAM)
DRAM	LPDDR1, DDR2	1 16-bit (LPDDR-400, DDR2-532, DDR3-800)	1 16-bit (LPDDR-400, DDR2-532, DDR3-800)	1 16-bit (LPDDR-400, DDR2-532, DDR3-800)	1 16-bit (LPDDR-400, DDR2-532, DDR3-800)
USB	3	2	2	2	2
EMAC	10/100	10/100/1000	10/100/1000	10/100/1000	10/100/1000
MMC/SD	3	3	3	3	3
CAN	1	2	2	2	2
UART (SCI)	4	6	6	6	6
PWM (Ch)		3	3	3	3
I2C	3	3	3	3	3
McASP		2	2	2	2
SPI	4	2	2	2	2

DMA (Ch)	32 Ch SDMA	64-Ch EDMA	64-Ch EDMA	64-Ch EDMA	64-Ch EDMA
IO Supply (V)	1.8, 3.3	1.8, 3.3	1.8, 3.3	1.8, 3.3	1.8, 3.3

From the above comparison chart we chose AM3358[4] processor for our implementation. As interfacing a SOC (System on Chip) is quite difficult we chose to go with a development board design. We again did some research on various development board designs available. Here is a comparison chart of various development boards:

Table 3: Development Board Comparison

Board	Raspberry pi	BeagleBone Black	BeagleBone	BeagleBoard-xM	BeagleBoard	TQ2440 S3C2440
SOC (Microprocessor)	Broadcom BCM2835	TI AM3359	TI AM3358/9	TI DM3730	TI OMAP3530	Samsung S3C2440A
CPU	700 MHz ARM1176JZF-S core	1GHz ARM Cortex-A8	720 MHz ARM Cortex-A8	1GHz ARM Cortex-A8	720 MHz ARM Cortex-A8	400-533MHz ARM920T
Memory	512 MB	512MB DDR3	256MB DDR2	512MB DDR2	256KB L2 cache	64M SDRAM
Interfaces	2 USB, UART, I²C bus, SPI bus , I²S audio	1xUSB,4x UART, LCD, GPMC, MMC1, 2x SPI, 2x I2C, 2xCAN Bus	1xUSB,4x UART, LCD, GPMC, MMC1, 2x SPI, 2x I2C,2xCAN Bus	2xUSB,McBSP, DSS,I2C, UART, LCD, McSPI, PWM, JTAG, Camera Interface	1xUSB,McBSP, DSS,I2C, UART, McSPI, PWM, JTAG	2xUSB, I2C bus, GPIO, camera interface
Video Output	Composite RCA, HDMI , DSI	microHDMI	microHDMI	DVI-D, S-Video	DVI-D, S-Video	DVI-D
Onboard Storage	SD / MMC / SDIO card	2 GB of eMMC flash memory	a microSD slot	4 GB microSD card	256 MB NAND Flash memory	256MB Nand Flash, 2MB Nor Flash
Onboard network	10/100 Ethernet	10/100 Ethernet	10/100 Ethernet	RS-232 port, Ethernet port	RS-232 port	100M Ethernet, 3 serial ports

For our implementation purpose we decided to go with Beaglebone Black[2]. As the design was industrially certified, it can be implemented safely in any of the embedded designs. We again made a thorough examination of the microprocessor

unit to make sure that all the requirements were met. It was successful in providing us with all the design requirements.

So, as for the circuit design for processing unit we chose to go with the Beaglebone Black.

Following are the concerned specification of Beaglebone Black[2]:
- ◦ 65 Digital I/Os
- ◦ 7 Analog Inputs
- ◦ 8 PWMs and 4 Timers (Analog Outputs)
- ◦ Ethernet and USB hosting capabilities
- ◦ 2 sets of Encoder Inputs
- ◦ 2 GB internal Flash memory
- ◦ Dedicated LCD Interface through Digital I/Os
- ◦ Intense processing capabilities (SOC clocked at 1GHz, 1 GB RAM) for analysis and Graphical representations.
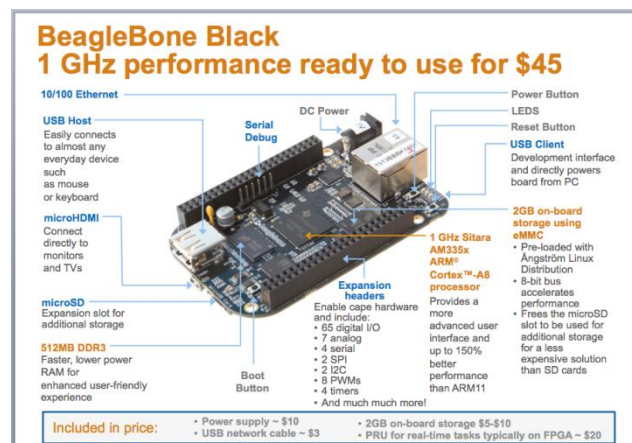
Figure 10: Beaglebone Black

5.2 Display Unit and GUI

After the decision for the processing board we started exploring the options for various options for our display unit. We firstly went with the designs that were compatible with the Beaglebone Black development board. For this few options were available such as:
- • Beadaframe
- • Beaglebonetoys LCD Cape[1]

- BeagleTouch Display
- BeagleLCD2 Expansion Board

We selected 7'' LCD touch screen from beaglebonetoys.com due to the following reasons:

o Highly compatible with BeagleBone Black.

o Large screen to cater our needs.

o Low power consumption

o No need for external daughter boards for interfacing.
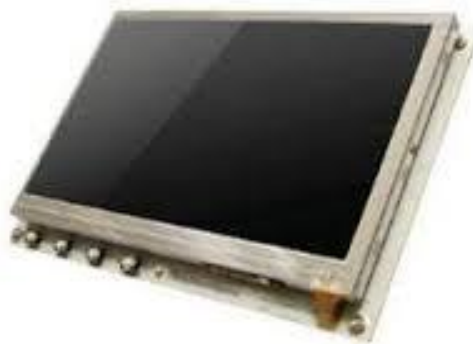
Figure 11: Beaglebonetoys 7" LCD

5.3 Interfaces

Among the various interfaces that were able Ethernet and USB were chosen due to following reason:

1. USB
   Widely accepted interfacing option that is available in nearly all mass storage devices. Being a popular Industrial Standard many of the processors support this including the selected TI Sitara AM3358. This will take of data logging activity as well as porting of programs and readings.

2. Ethernet
   Ethernet being a famous networking protocol has been implemented in various processor designs such as TI Sitara. As being a universal networking

protocol for computers, it can easily be used to connect with computers or on a network. It can also be used to control or monitor from a remote location on the network too.

5.4 Software Implementation

1) Platform

As for software implementation we chose to use Operating System on the system.

There are several merits over demerits of having a operating system rather than a firmware in complex programs and protocols. These merits have made impact in the implementation of complex algorithm in various embedded application.

Here is a discussion on merits and Demerits:

Table 4: OS Merits and Demerits

Merits	Demerits
Enables real-time, deterministic scheduling and task prioritization	Increased complexity
Abstracts away the complexities of the processor	Long boot time
Provides a solid infrastructure constructed of rules and policies	Power failure shut-down issues
Simplifies development and improves developer productivity	Testing and debugging complexity
Integrates and manages resources needed by communications stacks and	Interface responsiveness reduces due to background processes

middleware	
Optimizes use of system resources	
Improves product reliability, maintainability and quality	
Promotes product evolution and scaling	

After going through the table, we can clearly see that it is worth implementing on OS rather than a firmware.

Here are some arguments that support it:

▸ It will not only reduce the interfacing heads, but also help us to incorporate advanced features such as multithreading etc. that makes system more time efficient.
▸ The management of resources is unparalleled to any firmware implementation.
▸ Drastically reduces the development time due to abstraction it provides in implementing various resources.
▸ Provides libraries for GUI development and implementation of various complex tasks such as graphical representation[3] of data etc.

For the implementation we went with Linux based OS. As it is open source, so we can easily tamper with it through code. Also it is highly stable compared to windows, a required attribute for embedded system. Being free software, procurement cost is nil which makes it cost effective too.

2) Programming Language

As for the programming language we have decided on using Python[5] and its libraries (refer Appendix).

There were also programming language like C and C++ which are highly compatible with Linux core kernel but due to following advantages we decided to go with Python:

a. Excellent Documentation
   Python has got excellent documentation and tutorials on the internet that has resulted in boom in no of developers working on this language.

b. Easy to understand
   It has got very easy to understand syntaxes that even a non-programmer can easily understand. Its basic principle of syntaxes is governed by English language itself, which makes it very easy to understand.

c. Support for the Target Device (Beagle bone Black)
   Due to support present for python in Linux system, developers around the world are extensively using it to develop APIs to control its peripherals such I/Os, Timer, Communication channels and advanced inputs as ports to python.

d. Concise code
   It has got various inbuilt libraries and complex commands that saves developer from going into details of implementation. Due to this huge coding time has now been saved. Earlier Codes which measure up to 1 lakh line can now be coded in mere 4 thousand lines.

e. Object oriented language
   Being an Object Oriented Programming (OOP) language it has been easier to write codes in modular form which saves coding time as well as space.

   But still because of it being an interpreter based language it is rather slow as each line is converted to machine code at execution time. This could cause serious lags in the program that is not to be accepted in embedded designs. So in order to make it more efficient the code is now converted to machine code using Cython and GCC compiler. The code if firstly converted to C code with embedding a main function. After that we have compiled the code using GCC including python libraries. This compilation created a machine coded executable that can be used easily.

Still in order to make the code faster certain parts of the code has been coded in C++ and shell as they are closest to the Linux kernel. Parts of the code such as taking inputs from the encoder require a high sampling rate have been coded and compiled. Then it is used in the program by calling the compiled executable. That takes care of the lag.

But since creating a GUI is complex task we have coded that in python only that has been compiled to be executable.

3) GUI Development tool

As for creating GUI Screens we have used two tools:

1) GTK+3[2]

GTK+ (GIMP Toolkit) is a cross-platform widget toolkit for creating graphical user interfaces. This has been actively implemented throughout the python code. The GUI interaction with the python code was made possible with this widget toolkit only. It maps all the parameters as well as objects and brings them to python for it to interact and change based on the code.

Every action such as button click, text entry population of table and graphs was done through objects created using glade.

2) Glade[2]

Glade is a RAD tool to enable quick & easy development of user interfaces for the GTK+ toolkit and the GNOME desktop environment. This tool provides us a GUI environment to design GUIs. You can easily drag and drop various fields and place and set properties for various fields such as buttons, text entry, window, dialogs etc. The GUI screens can also be developed without it but due to following reasons it was chosen:

1. Rapid Application Development
   Due to its GUI interface it has been possible to make something around

40 screens. Without it there will be lot of coding to be done for creating and setting each attribute for every object. With this extent of work this tool has been a boon which nearly reduced the work ten folds without losing its functionality.

2. Reduction in executable code

As Glade creates all the developed GUI screens and save them in the form of XML file. We can easily interact with them and processing needs to be done at the time of execution of the code as they are coded in the form of XML document which are easy to execute then the coded ones. This in turns reduce the code as well as execution time to display screens. This drastically reduces the processing overhead on the embedded system making it efficient.

The above tools and approach helped me to develop an efficient program that can easily interact with user as well as hardware. The use of the above approach helped me to create a concise, readable, effective and reliable code for the program.

6. Drawings

1. Digital Interfacing Board
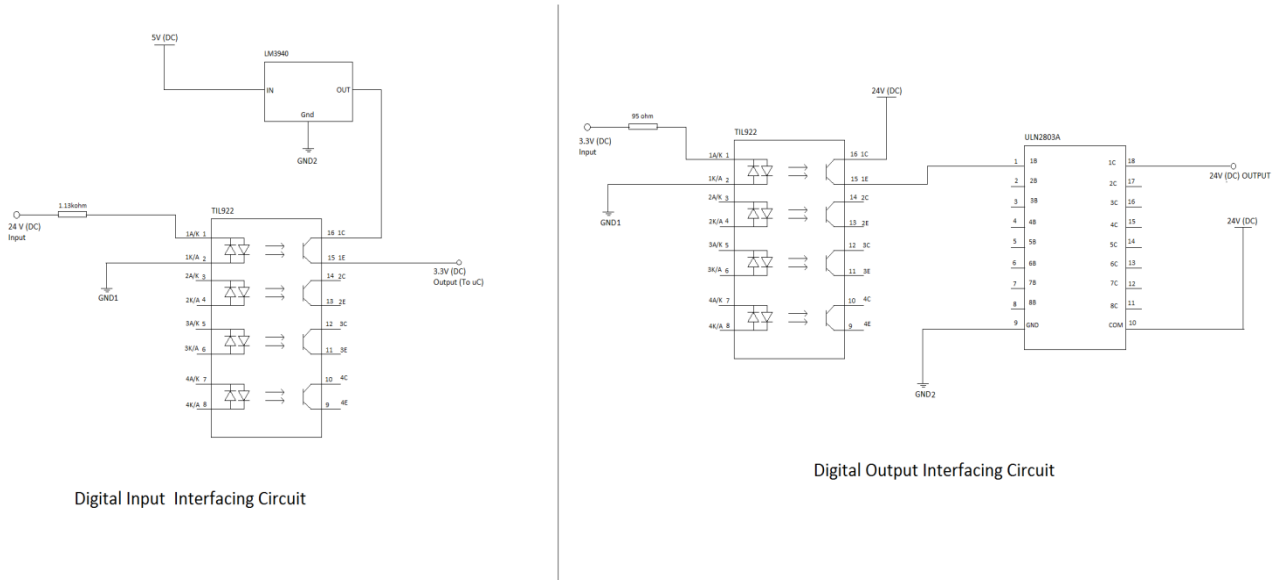
Figure 12: Digital Interfacing Board

2. Analog Interfacing Board
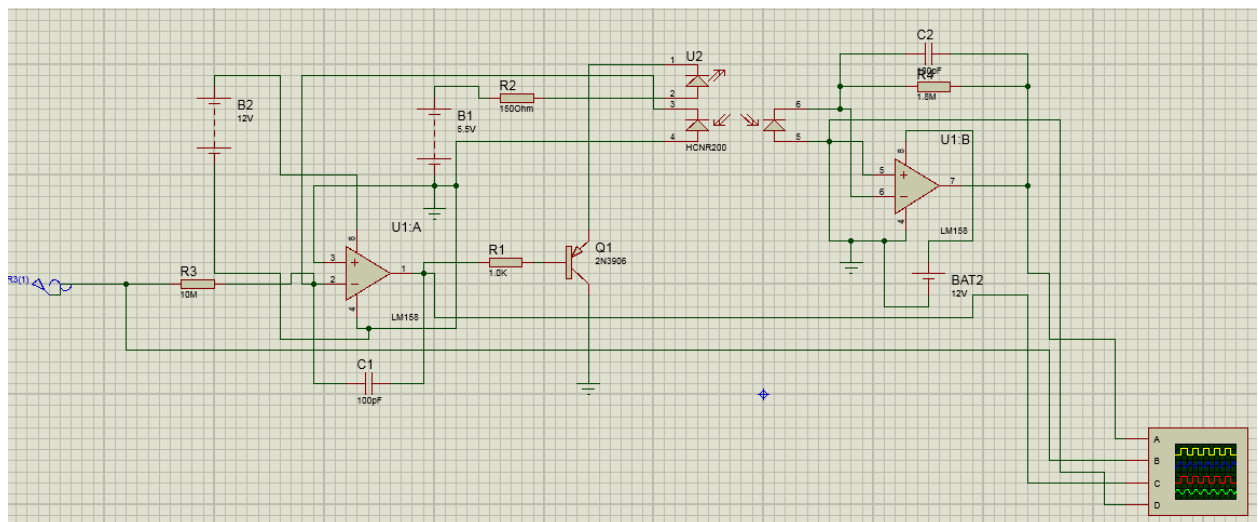
Figure 13: Analog Interfacing Circuit

3. Overall Circuit Drawing
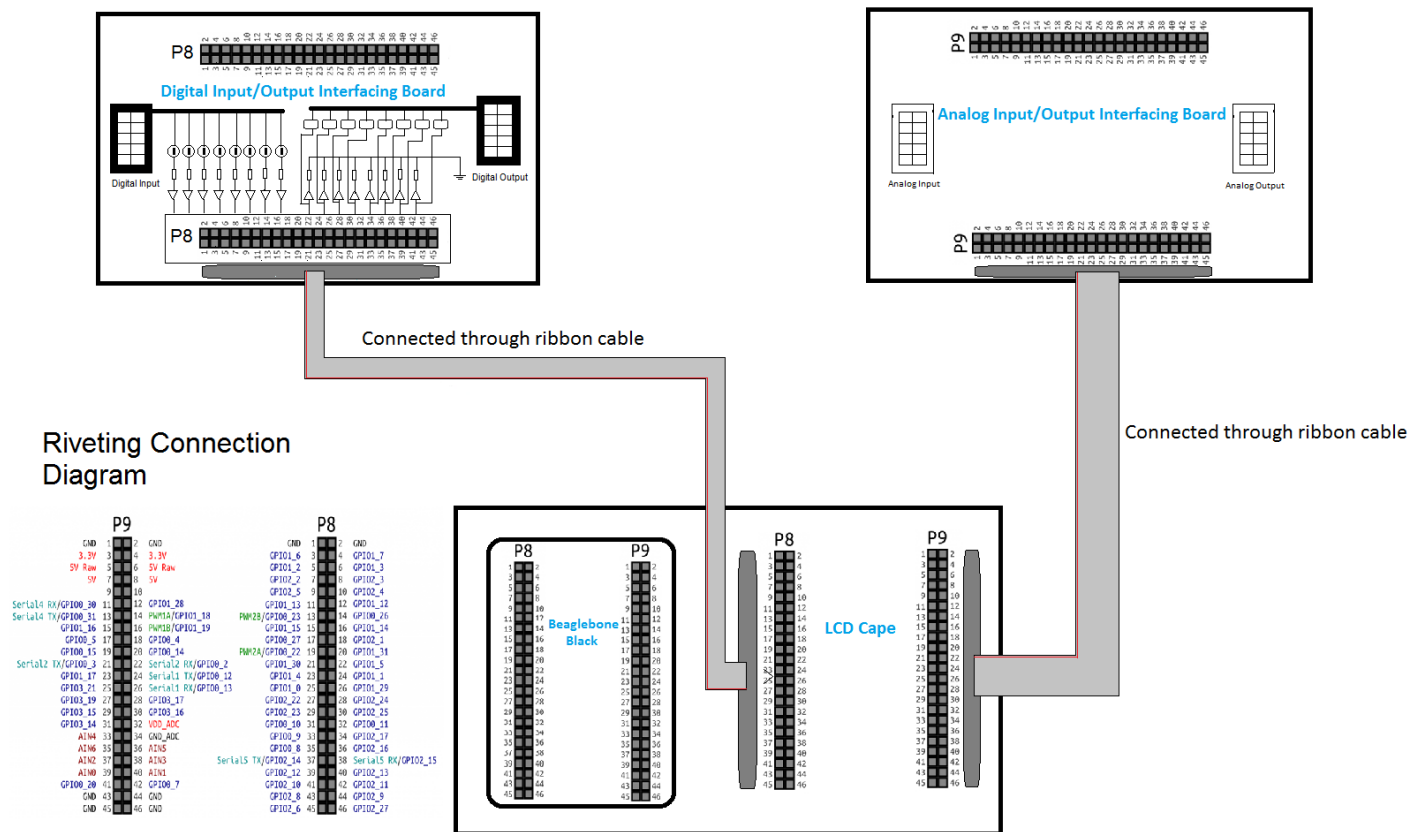
Figure 14: Overall Circuit Design

7. Process Flow

7.1 Defining the scope

Firstly we went with the evaluation of the requirement. Main requirements were:

- ◦ 26 Digital I/Os

- ◦ 2 High speed inputs for encoder input

- ◦ 2 Analog Outputs

- ◦ Data logging

- ◦ LCD based HID

- ◦ Analysis and Graphical representation

- ◦ Porting of data files to computer through Ethernet and USB Drives

The requirement was given for 26 Digital I/Os. Though clarity was not provided as to why these many I/Os were required. Actual riveting machine generally have a homing sensor, work piece sensor, encoder input, pressure sensor, motor control I/Os and valve I/Os. They didn't some up to 26 Digital I/Os. Still we went with those requirements and provided these many I/Os using shift registers in the Digital I/O boards.

As for Encoder inputs they come as a set of two inputs which were already provisioned in the development board.

Similarly analog outputs were provided from PWM pins available on the board other than LCD PWM pins.

Data logging was supposed to be for 68 programs but still we had lot of memory (i.e. 16GB) that we can use easily for each cycle. Generally each set of program files along with data and graphs were roughly around 2MB. So we can see that we can virtually store thousands of entries.

We chose for an LCD that is compatible with the pin layout of the board.

Also to save ourselves from the hassle of writing a driver for it, we decided to buy a compatible and supported LCD from beaglebonetoys.

Analysis and calculations were mainly done in C++ to improve on efficiency and reduce lags. As for graphical representation, we earlier had the dynamic graph plotting as the requirement but latter on we changed it to static one to reduce load on the processor while tabulating data. For plotting we used matplotlib in order to achieve those graphs as it is easy to implement. Manually plotting would have required for us to make another huge program for this purpose that wouldn't be even efficient. But as the price we couldn't do much on the graph sheet like drawing a box or area of concern, only graphs with legends was available so we went with that.

Porting of files was implemented through bash shell commands. This reduced our line of code to two or three lines. Also sensing a USB drive would have been very difficult without the help of shell commands as it requires to check within the root directories for mounted hardware. Still generating those shell commands was pretty time taking process, especially those for sensing the drives and mounting. Ethernet was supposed to be in the design but due to time constraints we abandoned that idea, as it required lot of research on FTP and Ethernet protocols.

7.2   Determination of Resources

Well after studying the whole requirement list we started with listing down our possible options for the design. We first started with our development board as explained earlier we went through lot of research to decide upon a development board. We not only studied various interfaces but also various processors that have been implemented in the design. After deciding on the development board, we went with decision for the display and HID (Human Interfacing Device). We decided on the Touch based LCD as discussed earlier. Then we went for deciding on the various interfaces that are required for machine like Ethernet etc. Then we decided on the power supply circuits and interfacing circuits. We devised circuits that were robust and meet industrial standards. Later we went with the software and programming language that was to be used. After making choices and decision for the various software tools that are to be used, we decided on a design for Graphics and User Interface. After having developed a rough sketch of the design we went with the Time vs Work evaluation.

7.3 Timeline Evaluation

After understanding various needs and development of the design, we went with the framing of a road-map, in order to get idea on various processes involved and amount of time required for each of the steps. Several steps such as designing and ordering of components have to be done simultaneously in order to work effectively as well as delays in project due to lead times in vendor management, ordering of components as well procurement of the components. Several processes like designing, programming and ordering went side by side. Well the plan was so, but due to companies internal procedures, some parts of the plan went to disarray, which led to halt of around a month. This significantly disrupted our work, because of which there was no time left for testing.

7.4 Major components of the project

Major components of the project was GUI design and development, programming for GUI engine and back end logic engine, designing of interfacing circuits and designing of power circuits. We learned a lot during designing of power circuits and interfacing circuits, as a person from not an instrumentation field, I faced a lot of problems during designing of circuits but thankfully my mentor did provided me with his experience and ideas that made me overcome this hurdle. Programming on the other hand went on for two months as various things needs to be studied and then implemented. After that GUI screen were developed that contained more than five hundred fields and buttons. As the collections of GUI screens were huge we went with a Rapid Application Development tool Glade which really reduced this huge work. Thanks to Glade we were able to develop the GUIs in a month.

7.5 Breaking down to smaller steps

Certain work such as programming and GUI development was too huge. It was getting difficult to even visualize as a whole. Because of this we weren't able to provide a clear view to our project managers. But then also we promised them to get it completed within the allotted time. So we divided the work of GUI development and programming to various stages. We first made development of GUIs as our priority, as to present some design related view to our project manager. We further divided the work into types of screens as process GUIs and settings GUIs. We firstly created the process

GUIs and after presenting them we went with the design of the setting GUIs. As for the programming we divided the work as GUI engine and backend Logic engine. As it was comparatively easier we went with that first. Though it was easier but the extent of the code was so huge, that it became really difficult to code the whole program into single file. So we coded the whole program in seven different files based on their functionality. After this huge task was accomplished we started with the development of the back end engine. We went through lot of troubles here. First trouble we faced was linking the program with the Operating System. Second was the file management system. Third was system state management system. We drafted for a design plan for a week as to how tom implement all this without affecting the other. After a week of hardship we finally devised methods and algorithms that can suitably and effectively work without causing any disruption to each other. Finally after two months of work, we were able to devise the whole back end system to manage everything.

7.6 Documentation

After this significant accomplishment we thought of noting down every bit of decision that we have taken during the whole development. Thanks to that we were able to tackle and answer each and every design related questions. We came to realize the importance of documentation. It didn't left astray whenever we were in a pinch, as all of our thought process was there on the paper we were able to look back, review and make required changes in the design.

7.7 Implementation

As for the implementation, we were supposed to do it one and a half month back but due to unavailability of components and company procedures, it got delayed to an extent that now we don't have the time to implement the design properly. So as for design showcase we are going with only some two basic modes. Also the testing device is still not ready as there is still some rework that has to be done on it. So looks like won't be able to actually test it on a real working machine.

8. Result

As for now designing of the system and circuits have been done. Also, programming has been completed and now we are waiting for the components to arrive. As for the result, we were able to produce only a prototype that is still not tested. The design was ready long before but due to time lag on the vendor's side, we weren't able to procure components in time. Also as the design has been outsourced, so not much of stress is now given on the prototype. It is still carried out as non-priority project just for academic purpose.

9. Discussion

The prototype design has been designed to meet all the features of a riveting controller.

It has all the data logging and necessary features but still lacks certain advanced things in its design. For example there is no choice for dynamic plotting i.e. plotting while acquiring values, self-detection of sensors and actuators and their health checkup, pinch zoom or zoom for graphs, display of concerned area in the graph. All these features cannot be incorporated due to programming overheads, loss of reliability and processing power.

Multitasking such as dynamic plotting was avoided as it can cause delay in execution in program which could lead to loss of accuracy.

As for the sensor and actuator detection we need a feedback from all the sensor and actuators meaning specific sensors and actuators that are compatible can only be used. We didn't support the thought as it will increase the expense and also exclusive for certain riveting machines.

Graph related problems such as pinch zooming and gesture controls are currently not available as a stable library in GTK, so we avoided it. Also only certain riveting modes are available due to lack of programming time.

But still for a basic riveting process this design quite reliable. It meets all the benchmarks for industrial scale controller. It has a basic and easy to understand interface that is quite user friendly to the operator.

So, as the final outcome, we can gladly say that design is quite robust and is equipped with many features that are sometimes not even provided in an industrial process controller. As it is a custom made design, we have thought of every helpful feature that should be there in the design

10. Conclusion and Recommendation

We currently are developing a prototype only. The actual design has been outsourced to TATA ELXSI due to lack of infrastructure for this kind of product development. Our purpose was to just check for the feasibility of the product, actual product will be done for Industrial Standards by TATA ELXI.

As for recommendation, I believe that we could have tried on the prototype a bit longer as there were no such deadlines to complete the project.

Also as the project is supposed to be indigenous product, we should have tried to complete the project all by ourselves. Instead of depending on outsourcing, we should have tried and be able to grasp some knowledge on this kind of product development.

New product development requires lot of effort, and patience from any individual or company new to this field. Generally product development takes 2-5 years to come into the market. We should have kept these things in mind and maybe would have opened a new market option for the company.

Appendix

1. Python

Python is a widely used general-purpose, high-level programming language. Its design philosophy emphasizes code readability, and its syntax allows programmers to express concepts in fewer lines of code than would be possible in languages such as C. The language provides constructs intended to enable clear programs on both a small and large scale.

Python supports multiple programming paradigms, including object-oriented, imperative and functional programming or procedural styles. It features a dynamic type system and automatic memory management and has a large and comprehensive standard library.

Like other dynamic languages, Python is often used as a scripting language, but is also used in a wide range of non-scripting contexts. Using third-party tools, Python code can be packaged into standalone executable programs (such as Py2exe, or Pyinstaller). Python interpreters are available for many operating systems.

CPython, the reference implementation of Python, is free and open source software and has a community-based development model, as do nearly all of its alternative implementations. CPython is managed by the non-profit Python Software Foundation. Nowadays, it has started moving into embedded systems too because of ease of programming and its base being C.

It is widely accepted language, so lot of documentation and product support is available

Libraries

Python has a large standard library, commonly cited as one of Python's greatest strengths, providing tools suited to many tasks. This is deliberate and has been described as a "batteries included" Python philosophy. For Internet-facing applications, a large number of standard formats and protocols (such as MIME and HTTP) are supported. Modules for creating graphical user interfaces, connecting to relational databases, arithmetic with arbitrary precision decimals, manipulating regular expressions, and doing unit testing are also included.

Some of the used libraries are:

1. Pygtk
2. Glade

3. Matplotlib

2. Pygtk

PyGTK is a set of Python wrappers for the GTK+ graphical user interface library. PyGTK is free software and licensed under the LGPL.

It is widely used in the development of GUIs for various python programs such as:

1. Anaconda installer
2. BitTorrent
3. Deluge
4. Emesene

5. Exaile

6. Ex Falso

We are using pygtk for development of a GUI for our programs.

Glade

Glade Interface Designer is a graphical user interface builder for GTK+, with additional components for GNOME. In its third version, Glade is programming language–independent, and does not produce code for events, but rather an XML file that is then used with an appropriate binding (such as GtkAda for use with the Ada programming language).

Glade is free software distributed under the GNU General Public License.

We are using Glade for designing our GUIs. It is easy to use, and provide lot many features that are difficult to implement via coding.
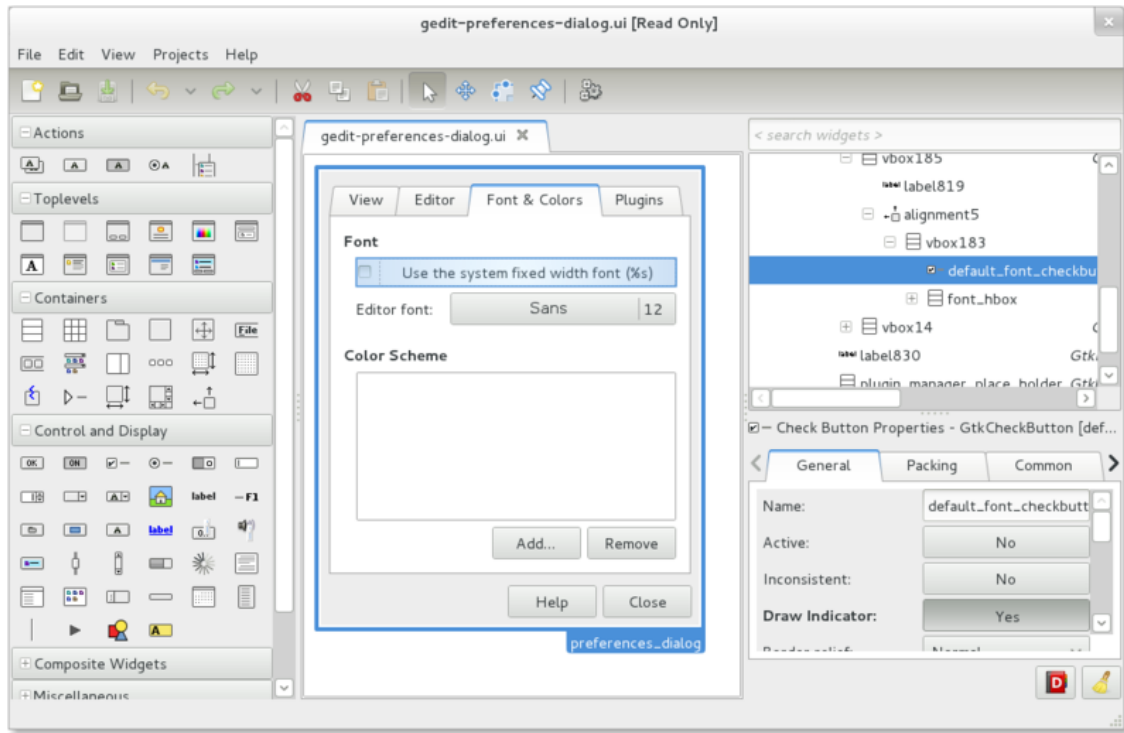
Figure 15: Glade Software

3. Matplotlib

Matplotlib is a plotting library for the Python programming language and its NumPy numerical mathematics extension. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like wxPython, Qt, or GTK.

Our basic implementation of 2D graph in the software will be done through matplotlib.
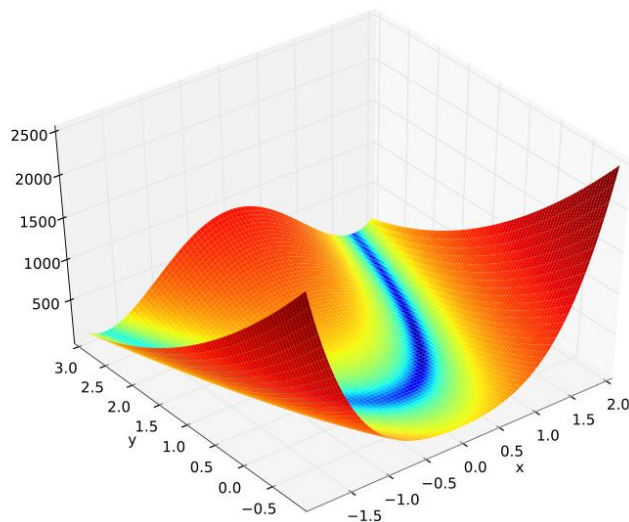
Figure 16: Graph plot by matplotlib

4. Program Screens
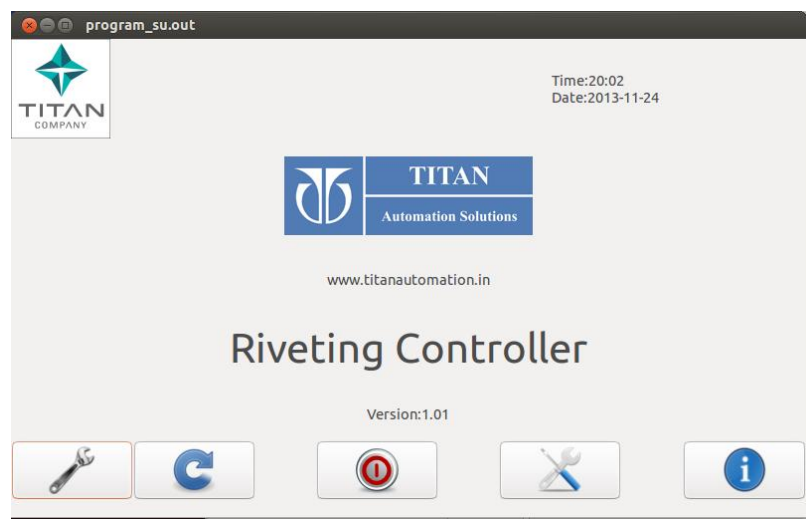
Figure 17: Main Screen
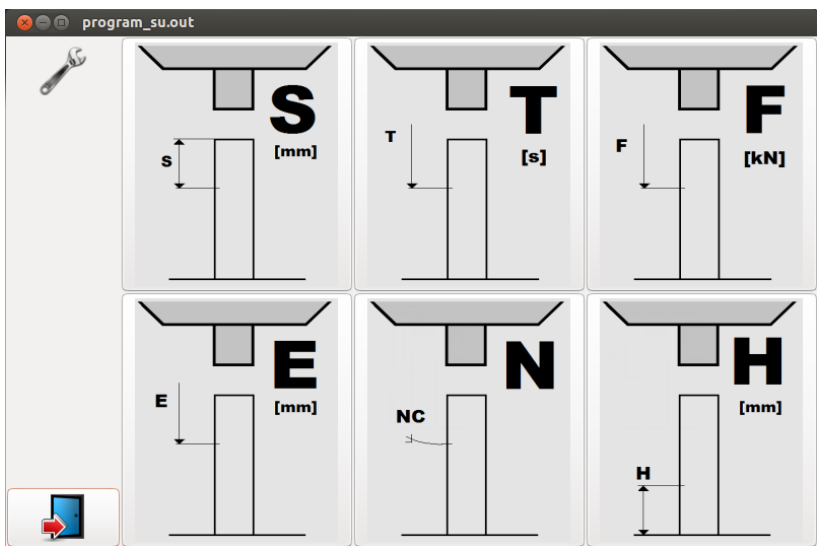
Figure 18: Mode Selection
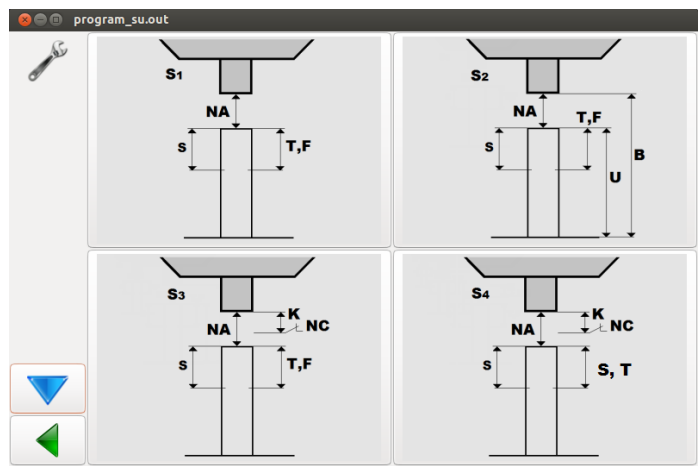
Figure 19: S1 Mode Selection
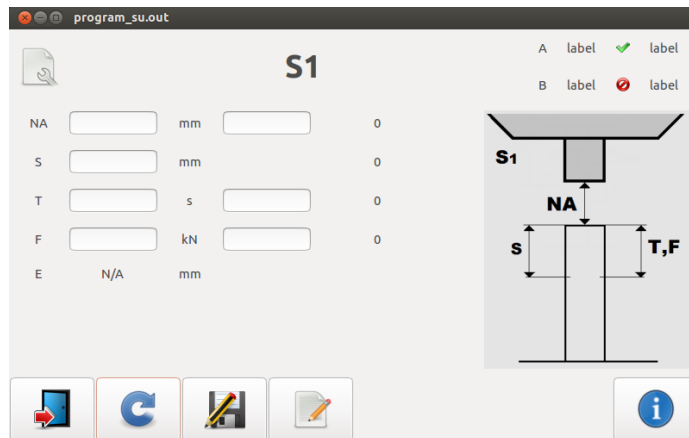
Figure 20: S1 Configure
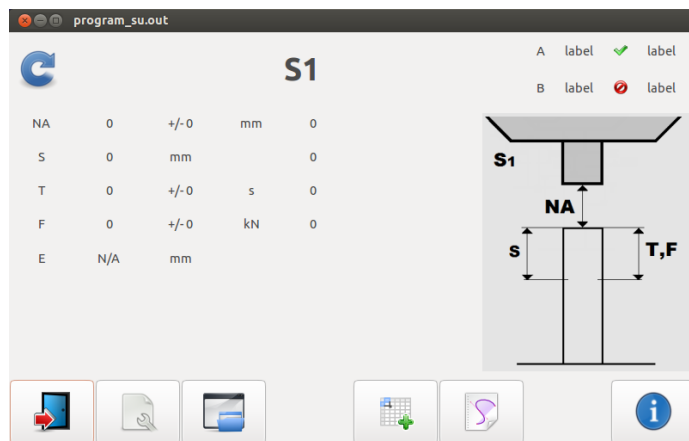
Figure 21: S1 Cycle
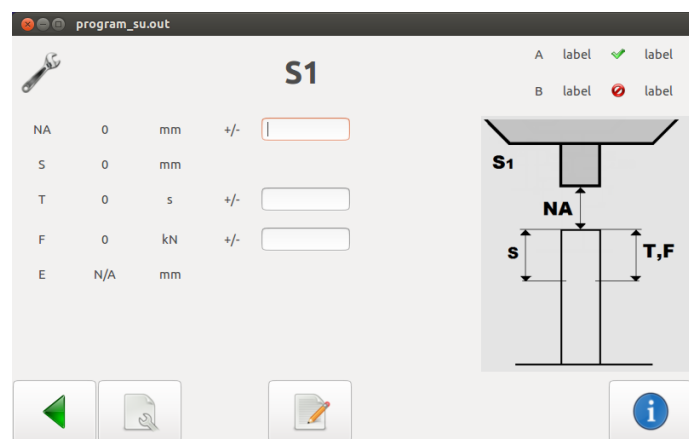
Figure 22: S1 Setup
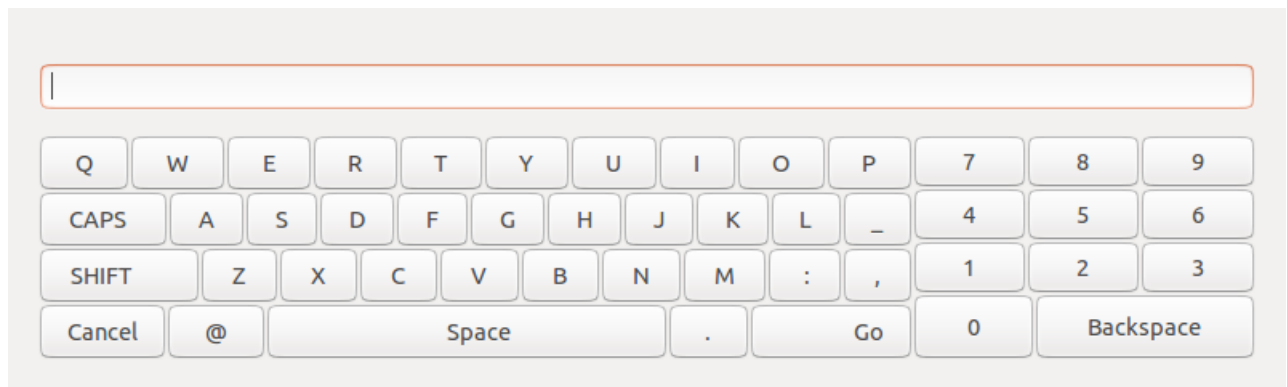
Figure 23: Keypad

Figure 24: Program Selection Dialog

Figure 25: Table

Figure 26: Graph
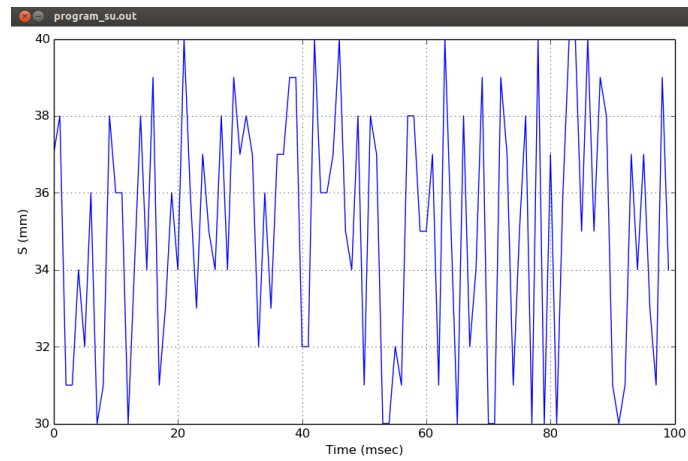
Figure 27: Graph Fullscreen
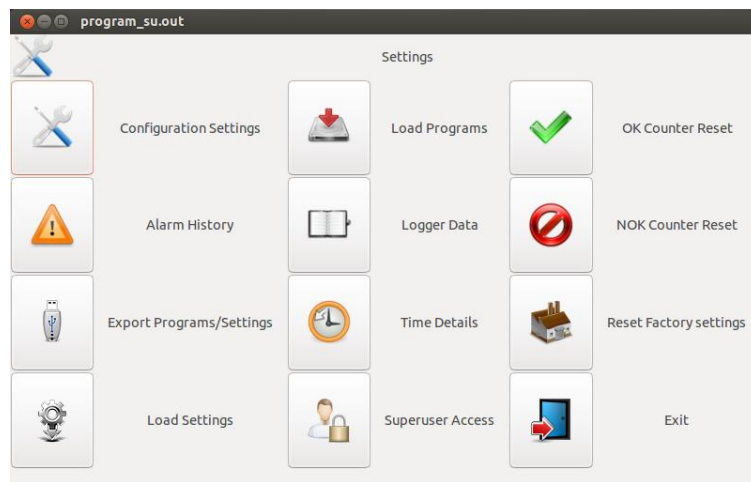
Figure 28: Settings
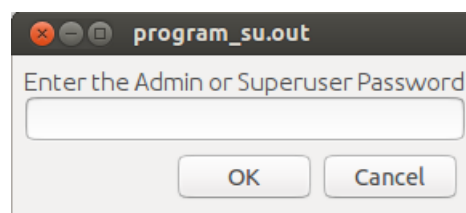
Figure 29: Password Dialog
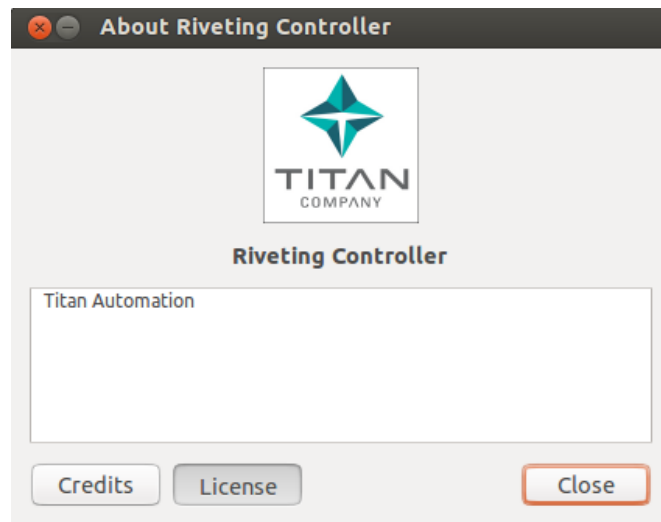
Figure 30: About Dialog
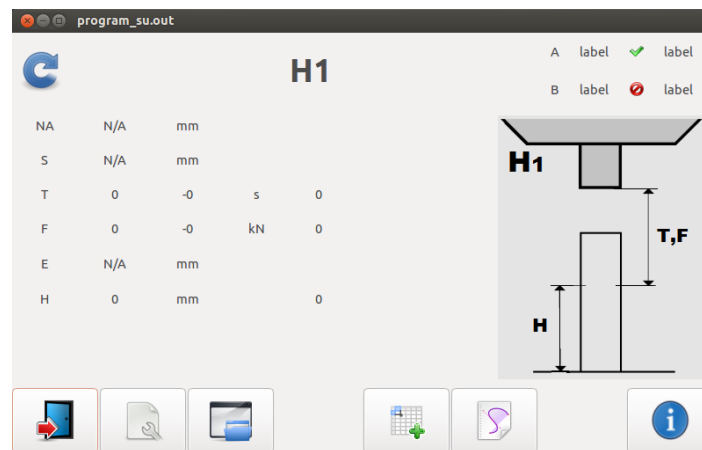
Figure 31: H1 Cycle
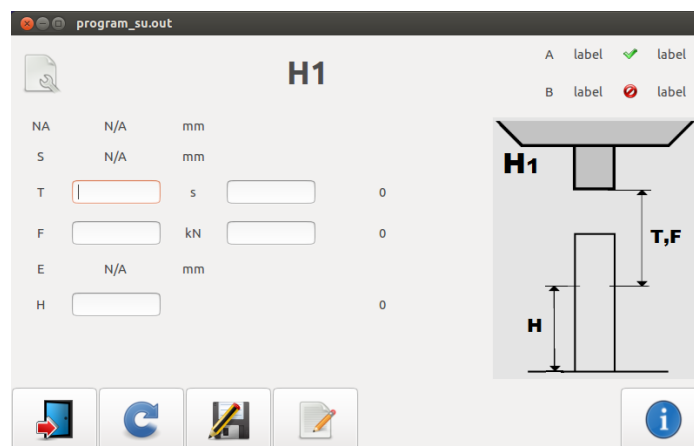
Figure 32: H1 Configure
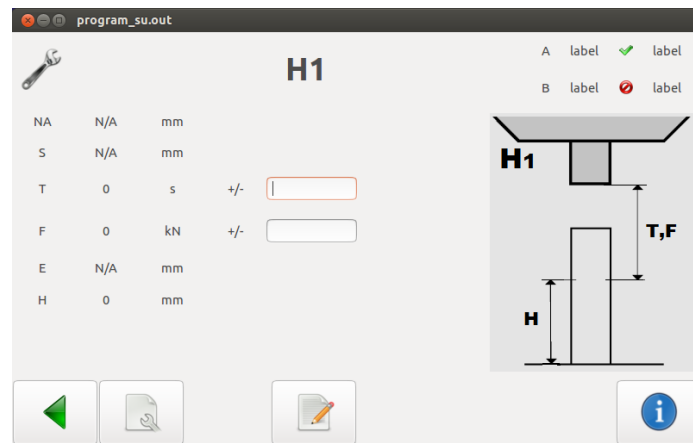
Figure 33: H1 Setup
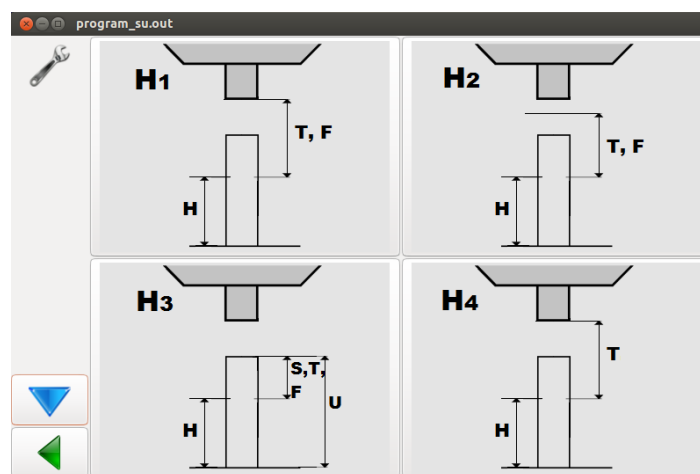
Figure 34: H1 Mode Selection
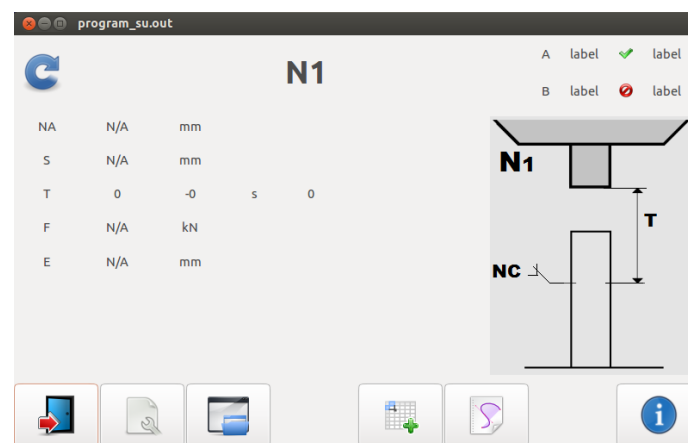
Figure 35: N1 Cycle
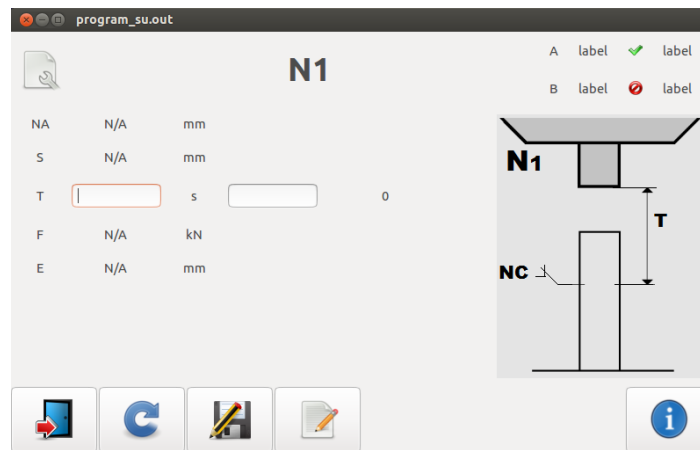
Figure 36: N1 Configure
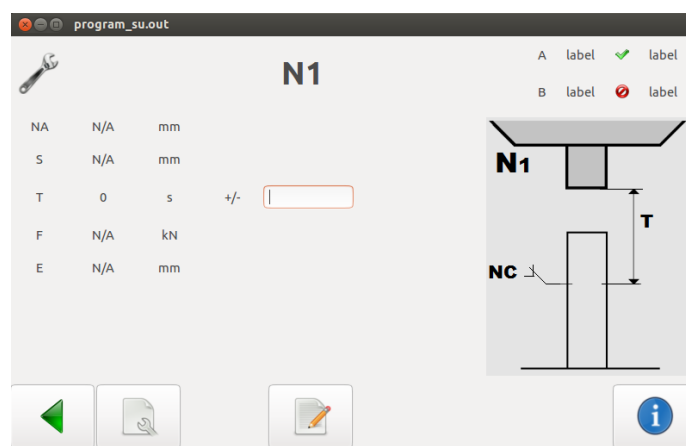
Figure 37: N1 Setup
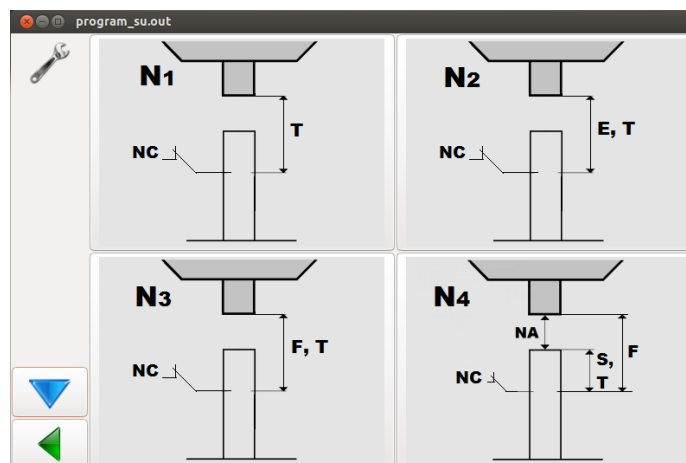
Figure 38: N1 Mode Selection
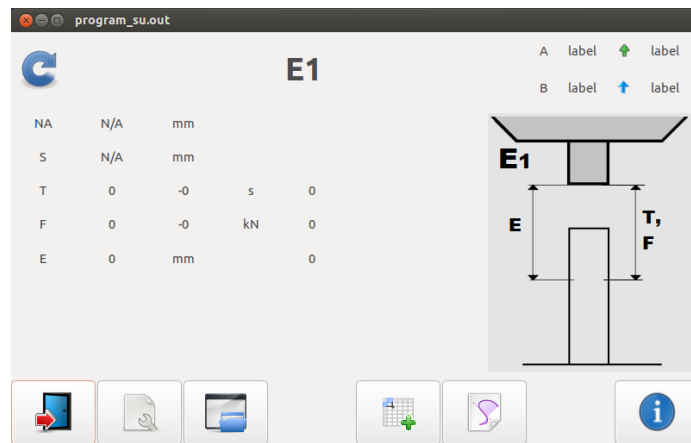
Figure 39: E1 Cycle
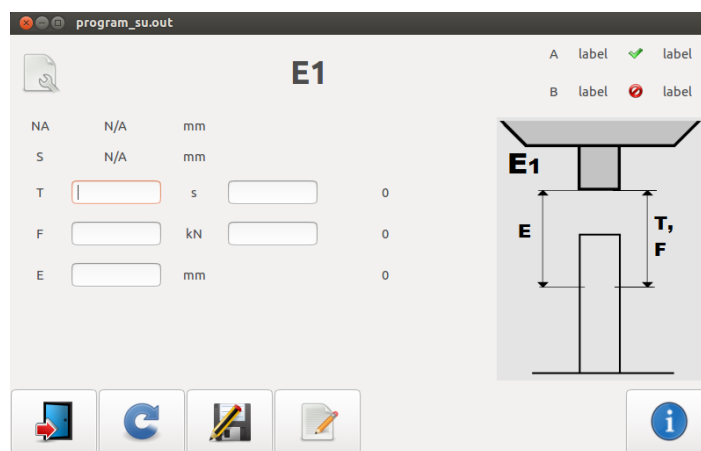
Figure 40: E1 Configure
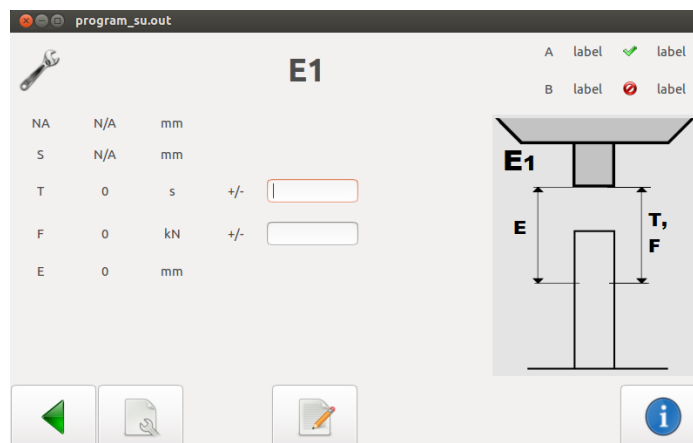
Figure 41: E1 Setup
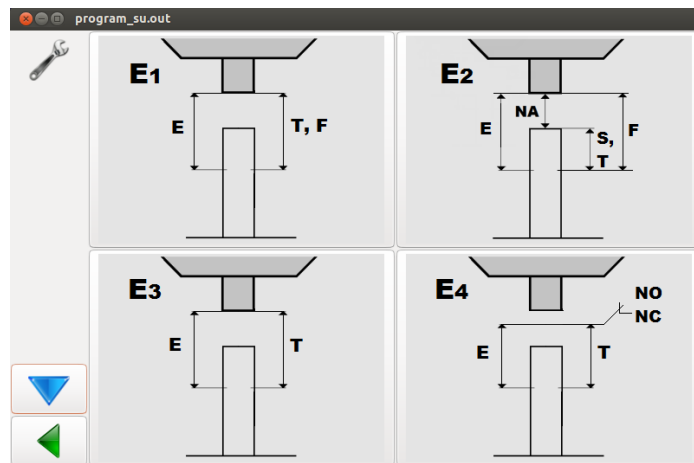
Figure 42: E1 Mode Selection
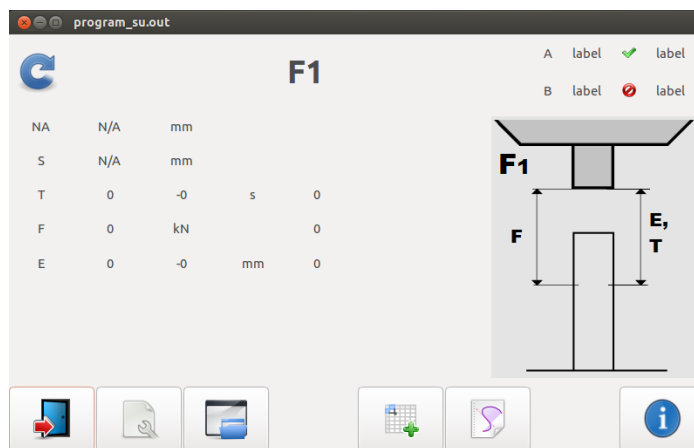
Figure 43: F1 Cycle
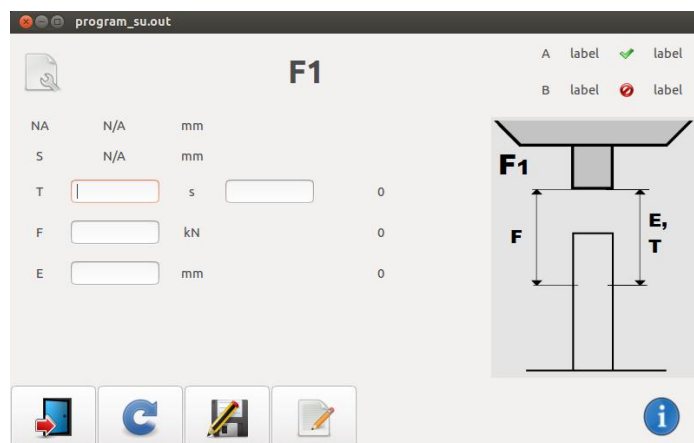
Figure 44: F1 Configure
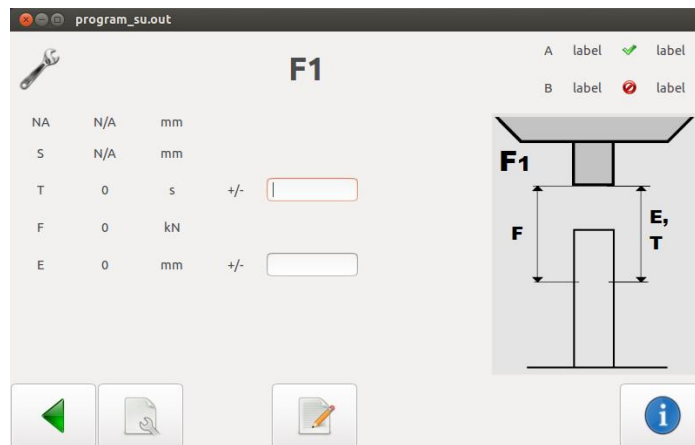
Figure 45: F1 Setup
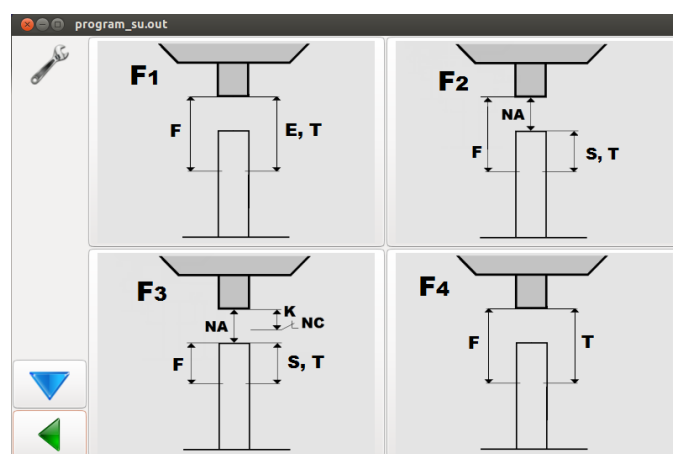
Figure 46: F1 Mode Selection
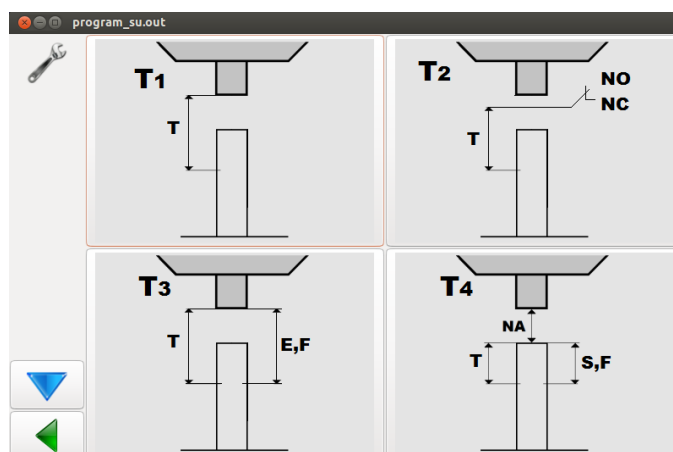
Figure 47: T1 Mode Selection
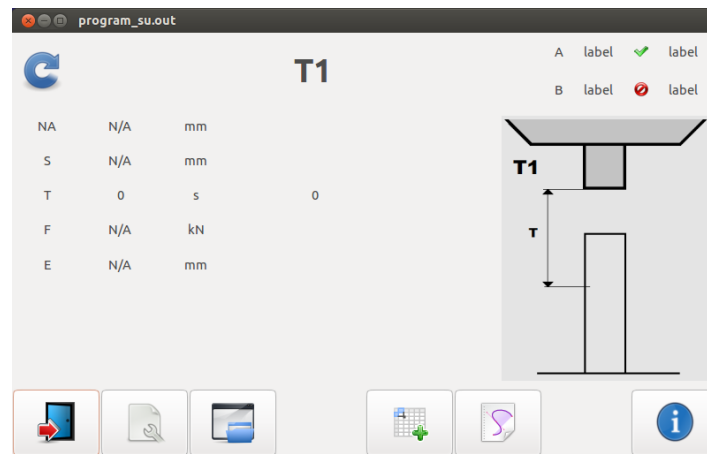
Figure 48: T1 Cycle
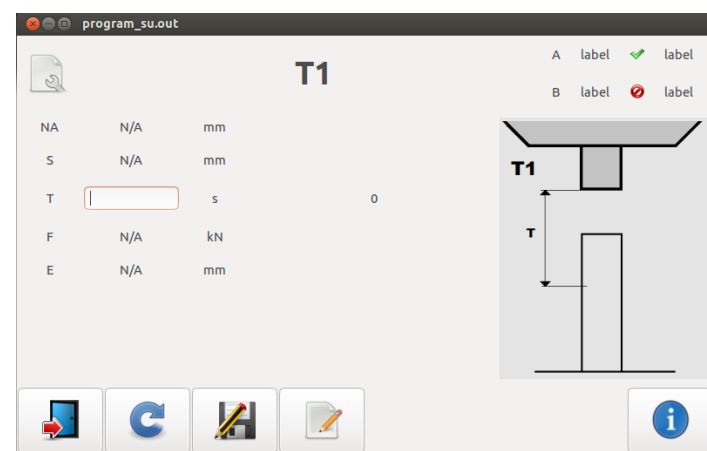
Figure 49: T1 Configure
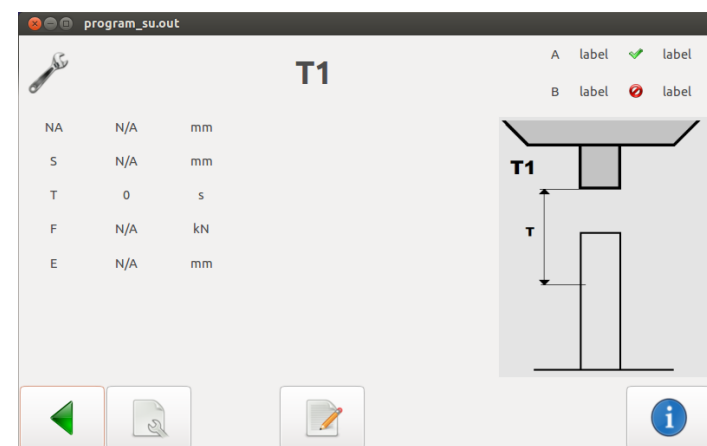
Figure 50: T1 Setup

Figure 51: Configuration Settings

Figure 52: Time Settings

Figure 53: Alarm

Figure 54: Advanced Settings

References

1. Beaglebonetoys 7'' LCD Datasheet
   (http://beagleboardtoys.info/index.php?title=BeagleBone_LCD7)
2. Beaglebone Black Datasheet
   (http://www.ti.com/lit/wp/spry235/spry235.pdf)
3. Texas Instruments ARM Sitara Am335x Datasheet
   (http://www.ti.com/lit/ds/symlink/am3359.pdf)
4. Texas Instruments ARM Sitara Am335x User manual
   (http://www.ti.com/lit/ds/sprs717f/sprs717f.pdf)
5. Python website
   (http://www.python.org/)
6. Google Groups
   (https://groups.google.com/forum/#!topic/beaglebone/dosKmEE7xso)
7. ST Microelectronics
   (http://www.st.com/web/catalog/sense_power/FM140/SC1798/PF69791)
8. Element14
   (http://in.element14.com/)
9. Wikipedia
   (http://en.wikipedia.org/wiki/Main_Page)
10. GIT Hub
    (https://github.com/Teknoman117/beaglebot/tree/master/encoders)