PHAN Pierre-Louis
1744U - GME2

AJAX Manufacturing
25 John van Niekerk St
Cape Town, South Africa

Stage de 2^{e} année
Cape Town, Afrique du Sud
Du **05/06**/2017 au **01/09**/2017

ROBOTIC AUTOMATION FOR GRAVITY DIE-CASTING

Summary

AJAX Manufacturing is a 60-years-old foundry based in Cape Town, South Africa. Its main activity is the **casting** of aluminium products, either specifically ordered by clients or directly sold by the company. The process of casting aluminium is permanently improved, and that is why I have been hired as an intern. The purpose of my internship at AJAX has been to **automate** the casting process.

To mass-product an aluminium piece by **gravity casting**, we have to follow a short cycle (30 to 90 seconds depending on the piece). A hydraulic circuit does the closing/opening of the **die**, and provides the force to clamp it together. In some cases, the geometry of the piece needs the use of a core, placed between the two halves of the die. Then the operator carefully pours the molten aluminium into the sprue. After the required cooling time, the die is opened, the core is removed and the piece can be taken out.

Many of the various aspects of the table could to be automated : **cycle management**, movement of the hydraulic cylinders, interface with the operator, aluminium pouring, **temperature control**, and safety. Each of these needed completely different types of work, from computer programming to precision mechanics. This required very diverse abilities ; most of which I did not have before, and learned either by asking others or watching them operate, or by looking online if nobody was able to help me. Some of the technical knowledge that I needed was linked to what we did during the school year. Working for a strictly professional purpose allowed me to refine the basis of mechanics and electronics that I acquired in the practical exercises and the PIER project.

AJAX Manufacturing est une fonderie Sud-africaine, fondée il y a 60 ans au Cap. Son activité principale est le **moulage** de pièces en aluminium, soit commandées par des clients, soit conçues et vendues directement par la société. Le procédé de moulage aluminium est constamment amélioré, et c'est pour cela que j'ai été recruté comme stagiaire. Le but de mon stage à AJAX a été d'**automatiser** le procédé de moulage.

Pour produire en série un pièce, en **coulant** de l'aluminium, il faut suivre un court cycle (de 30 à 90 secondes, en fonction de la pièce). Un circuit hydraulique s'occupe de fermer/ouvrir le **moule** et de le serrer. Selon la forme de la pièce à couler, on ajoute parfois un noyau entre les deux parties du moule. L'opérateur verse ensuite l'aluminium liquide dans l'ouverture supérieure. Après le temps de refroidissement, on ouvre le moule, on retire le noyau et on peut extraire la pièce.

Plusieurs aspects de la table de moulage ont pu être automatisés : **gestion du cycle**, mouvement des vérins hydrauliques, interface avec l'ouvrier, coulage de l'aluminium, **contrôle thermique**, et sécurité. Pour chacun de ces aspects, le travail demandé fut très différent, allant de la programmation informatique à la mécanique de précision. Cela a demandé des compétences très variées, que pour la plupart je ne possédais pas. Je les ai acquises en demandant l'aide des ouvriers ou en les regardant faire, ou si ce n'était pas possible, en me renseignant sur internet. Certaines connaissances techniques dont j'ai eu besoin m'ont ramené à ce qu'on avait fait en école pendant l'année. Travailler dans un contexte strictement professionnel a permis de fortement approfondir les bases acquises en travaux pratiques et en projet PIER, dans les domaines mécanique et électronique.

Table of contents

The company : AJAX Manufacturing and the facilities 3

Table Automation

Preamble 5

1. Study of the Casting Process

1.1. Study of the Cycle 6

1.2. Study of the Temperature 8

2. Solutions - Hardware

2.1. Motion of the Cylinders 11

2.2. Temperature Sensors 13

2.3. Human-Machine Interface (**HMI**) 14

3. Solutions - Software

3.1. PLC General Program 15

3.2. Temperature Acquisition 18

3.3. Graphical Interface 19

Conclusion 21

Appendix A 22

Appendix B 27

Synthèse en français 31

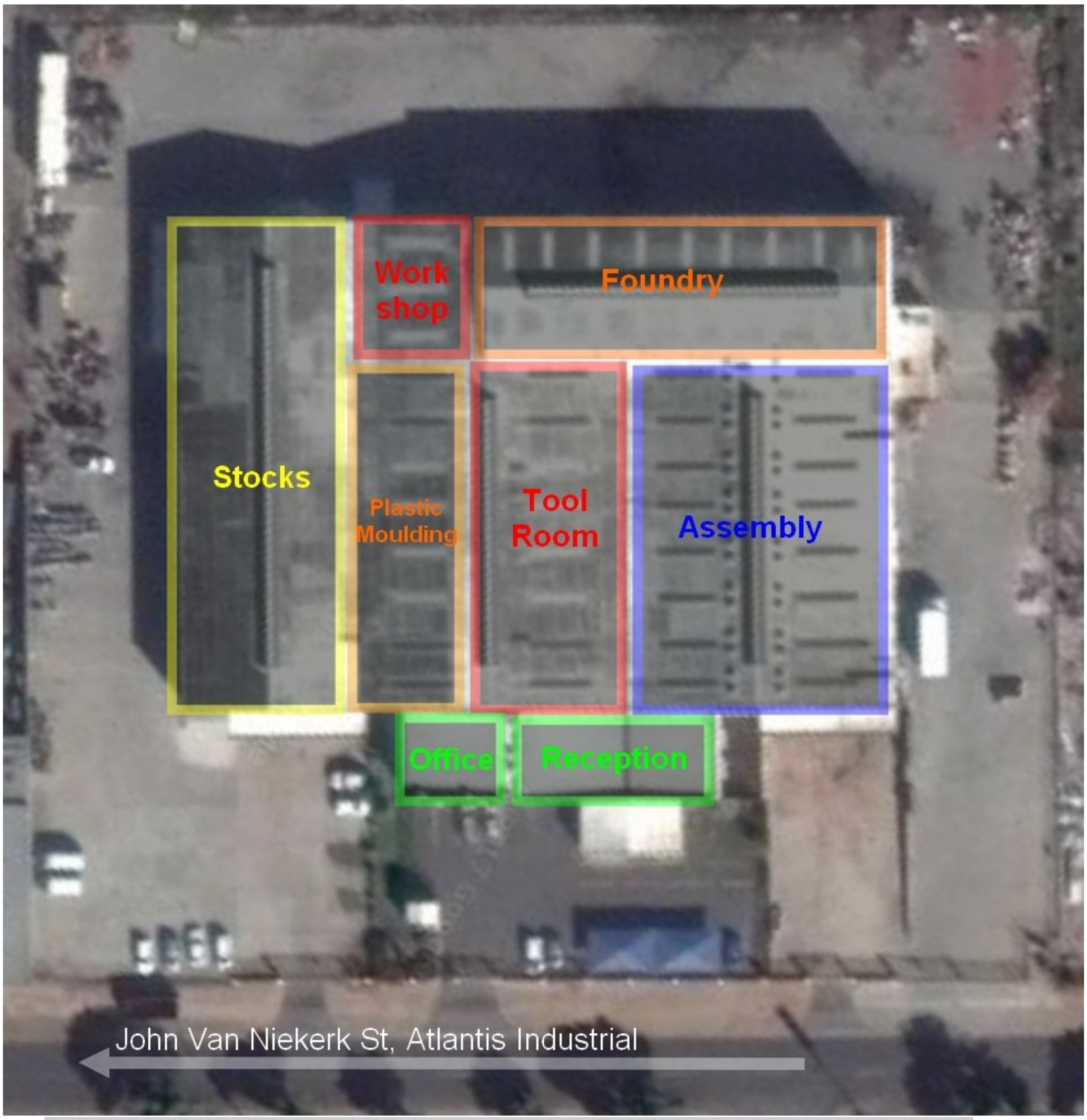
The company : AJAX Manufacturing and the facilities

AJAX Manufacturing is a foundry founded in 1951 in Cape Town, South Africa. It started doing **aluminium gravity casting**, which is still today its main activity. Since joining Sicame Group in 2010, AJAX widened the range of its activities : solution design, tooling, machining, plastic injection and assembly.

The products (aluminium and plastic) are either specifically ordered by clients or directly sold by the company. In both cases, the design and the making of the die are the critical parts of the process ; after that, the step of mass production can be started. After being cast the piece is often treated (sanding, heating) and also assembled, so that they sell products that are ready to use.

There are only a dozen foundries in the Western Cape province, racing for the most advanced techniques in metal casting. In the context of a fast developing country, innovations in industrial procedures are more and more possible, and there is much room for improvement.

The three months I spent in the company were enough for me to witness some changes in the general organization, the quality management and the safety measures, which I will discuss later.



⇧ *AJAX facilities. It is divided into parts corresponding to the different activities.*

The **offices** are where everything is organized : direction, management, purchases, human resources, quality management, and the department where I worked : research and development.

The **foundry** is where the aluminium is molten and cast, and the aluminium pieces are machined and treated. Obviously the first part of my work was to observe the whole casting process, in order to draw conclusions on what could be improved, what could be automated, and how to do it.

The **tool room** is equipped with various machines (CNC or hand-operated), to make the pieces and the dies designed by the engineering team. It has milling/drilling machines, CNC routers, lathes, bandsaw tables, grinding and sanding machines. This is where I was building the casting table, putting the components together.

The maintenance team is there to help anyone, on technical problems. They have their tools in the **workshop** and work there occasionally, but they are mostly dispatched wherever they are needed. I worked with them a lot, and their knowledge in tools, electronics and machines allowed me to learn quickly at the beginning of the internship.

⇧ *The tool room*

⇧ *The foundry*

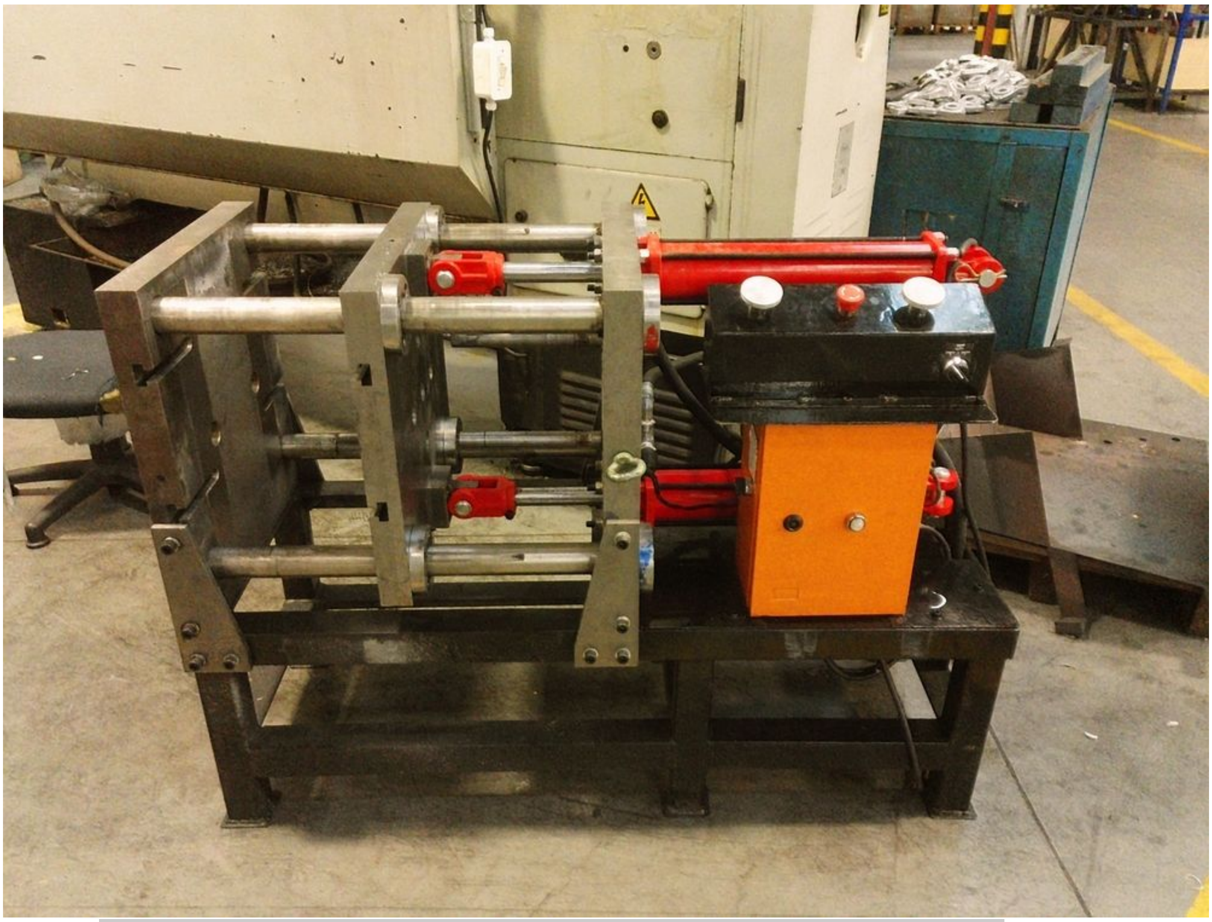
The **plastic moulding** activities of Ajax are similar to the aluminium foundry, except that the liquid plastic is pressurized and injected in the dies. The whole process, from the injection to the extraction, is done automatically, which is easier to do than with aluminium.

The different plastic parts then go to the **assembly** where employees put them together, sometimes also with metal parts, to make the full product. It is finally sent to the **stocks** departement, put into bags that are weighted and sealed, to be ready for shipment.

Table Automation

Preamble

The concept of a casting table has different variations. In all cases, it has a fixed frame supporting two fixed plates. Another plate with 1 degree of freedom can move back and forth in between, opening and closing the die previously set up in the table. The existing table was in the most **basic mode**, with two buttons : keep pressed the right button to open the die, keep pressed the left button to close the die.



⇧ *This is the casting table I was working on. No die is set up at this moment.*

Making this process automatic, so that the worker only pours the aluminium while an **automaton** handles the closing/opening of the die, brings some important benefits :

- It increases **consistency**, so the quality of the pieces is more stable
- It imposes a **pace**, and a number of pieces made per table per hour

The first step to achieve this has been to **study the casting process** in detail, to determine what solutions I would have to implement. Next, the setup of all the **hardware components** ; and then the **programming of the automaton** and the digital screen.

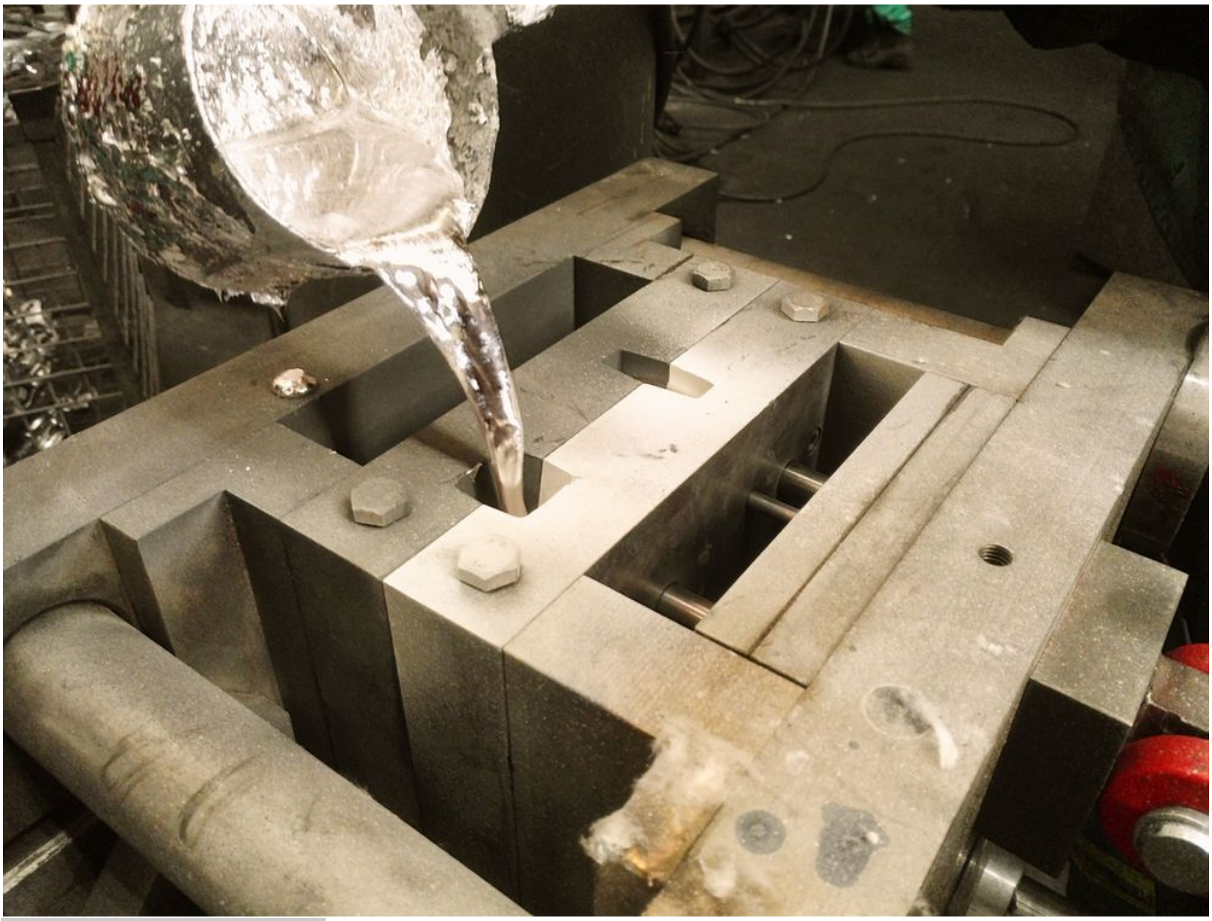
1. Study of the Casting Process

1.1. Study of the Cycle

Doing the aluminium casting with the employees allowed me to know the exact **sequence of the movements** of the middle plate, during a full cycle.

In theory, for the automatic mode, the actions to engage are simply '*fully opening the die*' and '*fully closing the die*'. But the first thing I noticed is, the theory rarely corresponds to the real going of the operations : in addition to these **theoretical movements**, we need to engage some **additional movements** for diverses reasons ; here is the **full cycle** in detail :

a. First of all the die must be heated to its **starting temperature T_0**. We **open the die slightly** to have a small gap (5cm to 10cm) and place a blowtorch between the two halves.
b. We must then spray the **coating** on the interior surfaces, so we **open the die more**. The die being completely opened, and liquid being sprayed on the surfaces, a lot of heat is lost during the coating. The die has to be heated again with the blowtorch, so we must close the die and leave the small gap to let the flame heat the interior surfaces, until the starting temperature T_0 is reached again.



⇧ *Aluminium being poured.*

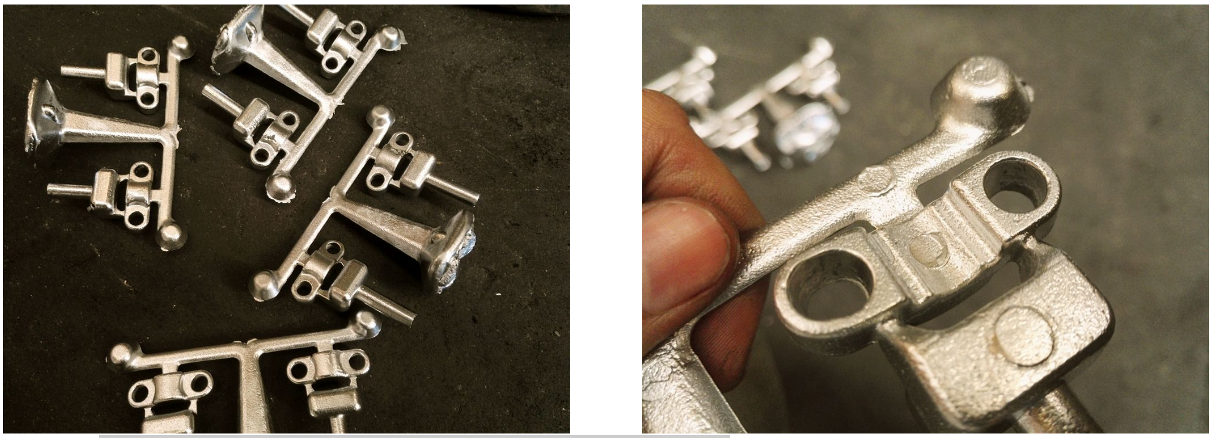
We can see the dark coating on the left part, and the bright coating on the right part

c. Now we can start casting the pieces, so we **close the die completely** to pour aluminium into the duct.

d. After the cooling time, the aluminium has solidified, so we have to take the piece out of the die. In theory when we open the die, the piece is supposed to **break off from the fixed half** of the die, and stick to the mobile half. Most of the time, some parts of the piece stick to both halves. As soon as it happens, we have to **close the die** again and **reopen it**. We have to repeat that process (usually 1 to 3 times) until the piece breaks off from the fixed half.

e. Then we **open the die completely**. When the die is fully opened, two **ejector rods** come press the back of the die, it makes the **ejector pins** push the piece out of the shell. We can now take the piece out.

f. Finally, we have to **close the die completely** to push back the ejector pins inside, and the cycle is started again.



⇧ *Results of the castings, and close-up on a piece.* ⇧

The main issue with a fully automatic mode is that it cannot deal with unexpected elements, like whether or not the piece breaks off easily from the die (and from what side?). This is where the **coating** plays an important part. Two sorts of coating are used simultaneously. They are both sprayed on the surfaces which will be in contact with aluminium, but they serve different purposes :

- The white coating is there to help the heat transfer from the liquid aluminium to the die. The faster the molten aluminium is able to loose its heat, the faster it solidifies ; so it reduces the cycle time.
- The black coating prevents the die and the piece to stick together. We spray it only on the fixed half of the die, so that the piece sticks to the moving side. Once the die open, the absence of black coating on the moving side is not a problem, because the ejector pins safely break off the piece from the die.

When we follow this cycle continuously, the first few pieces (5 to 10 pieces) will not be acceptable. They go directly to the **scrap pile**, with all the pieces whose quality does not meet the expectations. It is because the starting temperature T_0 obtained with the blowtorch is not high enough. It takes a few castings to heat it up completely, the temperature of molten aluminium being higher than T_0.

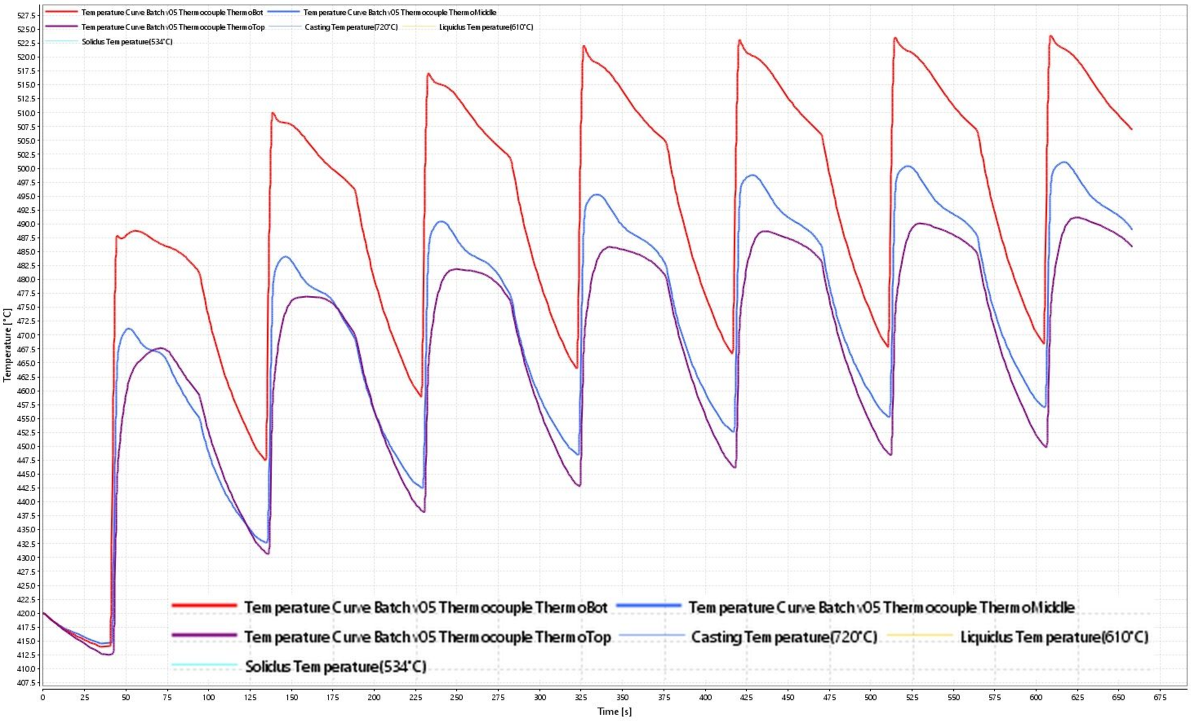
The next chapter describes this evolution of temperature after starting the castings.

1.2. Study of the Temperature

Temperature control is not a *necessary* aspect of gravity casting : the resulting quality can be sufficient without it. However, during a process involving the fusion and solidification of a material, it is clearly better to operate knowing the current temperature of the system.

The goal here was not to *control the temperature*, but to allow the automaton to know what step of the cycle we are in, according to the live temperature. For that I began by studying the behaviour of the temperature curve, as we do castings.

We cannot read the temperature of the aluminum itself, neither the temperature of the surface of the die : the measuring device would obviously alter the shape of the piece. But we can read the temperature inside the die. As the same cycle is repeated, the resulting temperature is a **periodic** value. Since heat diffuses within the metal, the further away from the surface we measure the temperature, the flatter is the temperature signal. So we need to measure the temperature as close to the surface as possible.



⇧ *Here are the results of a simulation, done for a piece actually produced by Ajax*

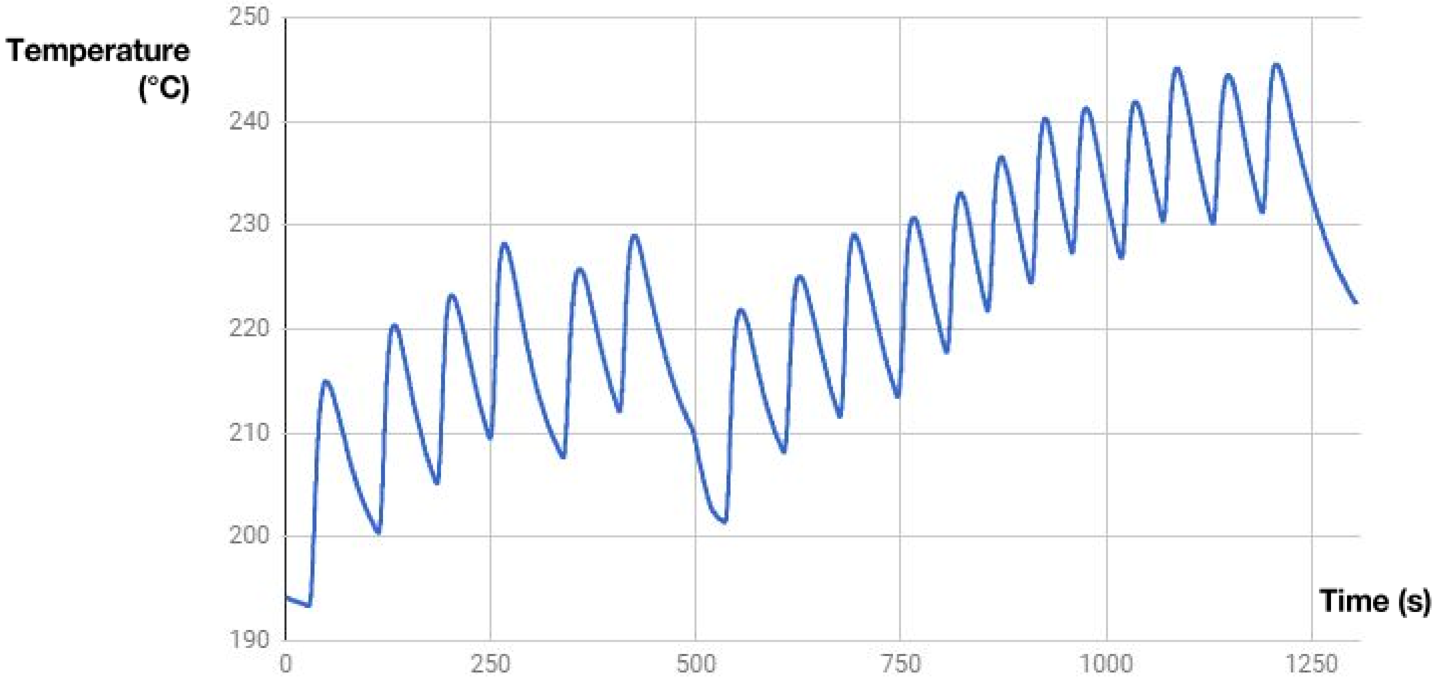
On the simulation we can see the different temperature curves corresponding to different distances to the surface : the closer to the surface, the wider is the amplitude of the signal. However, for the three curves, we observe the same pattern :

- At the scale of one cycle, there are 3 distinct moments:
  a. An abrupt **rise of temperature** caused by the addition of molten aluminium. The amplitude is typically 60°C if closest to the surface, 45°C if further away.
  b. A **slow decrease**, corresponding to the cooling by ambient temperature. This is when the piece solidifies. Typically 15°C.
  c. A **fast fall**, happening when we open the die.

- At a larger scale, we clearly see a global trend.
    a. In the beginning, after one cycle the piece does not cool down to the initial temperature ; so the next cycle begins higher than the previous cycle did. In average, the die is warming up. This is a **transitional state**, when the resulting pieces do not conform to the expected quality. We observe defects like shrinkage or gas porosity, and the pieces are tossed to the **scrap pile**, to be remelted.
    b. After a few cycles the average temperature increases less and less, and at one point (5 to 10 cycles) reaches a stable level. This is the **steady state** which results in pieces of **consistent quality**. If everything goes well, it lasts until we pause production. To keep this as long as possible we need to be consistent in the process. (The automation makes that easier, but there is two operations where the pace is still up to the human operator : pouring the aluminium, and taking the piece out of the die.)

Now we have a good idea of the behaviour of the temperature curve, but this is still a simulation. For the next step I studied the real evolution of temperature. I had been given an outdated die, to experiment on, and figured out how to build the sensor on it (more details in the chapter about hardware : 2.2. Temperature Sensors).

We started doing tests in the foundry. The workers did not like the idea of doing castings “just for tests”, knowing that the resulting pieces were useless, and going to be remelted. But they helped me anyway, and ended up really interested in seeing the live temperature curve displayed on the computer.



⇧ *The data of the first test.*

The first thing to notice is that, although the ‘small scale’ model (rise of temperature, slow decrease, fast fall) is not verified, we can see each **rise and fall** separately. In addition to this, the ‘large scale’ model (**transitional state, steady state**) is indeed here. The interruption at around 500 seconds, after 6 castings, is only because we sprayed some more coating.

From these results, I decided to consider **3 key temperature values** :

- First the automaton needs to know when we start the castings. This one has already been mentioned : the **starting temperature T_0** is the temperature to reach with the blowtorch. Once at T_0, we start the castings.
- Then the it needs to know when to open the die, to let us take the piece out. It is not a constant temperature value : from each peak, the temperature must decrease of a certain amount, then the automaton opens the die. It is a **delta of temperature ΔT**. From each peak T_{max}, the die opens at $T = T_{max} - \Delta T$.
- Finally the automaton must know when the steady state is reached, when we can keep the resulting pieces. This is the **target temperature T_1** : at each peak T_{max}, the automaton tests if $T_{max} > T_1$, and advises.

First problem encountered : for each different die, the key temperatures will differ. In the case of the die I used for the tests, we can choose these values using the graph above : T_0 = 194°C, ΔT = 12°C, and T_1 = 243°C. But for the automaton to work with every die, it needs to be told what are its key values. For that, I used the HMI (see chapter about hardware : 2.3. **H**uman-**M**achine **I**nterface). In addition to the screen, it has buttons and a numeric keypad, that I programmed for the operator to be able to enter these values.

Second problem encountered : the temperature acquisition is late behind the real temperature of the surface of the die. This delay is due to the distance between the sensor and the surface. It does not prevent the detection of T_0 and T_1, because those are slow evolutions. However it is a problem for the detection of ΔT. The rise of temperature happens within 5 seconds, which is roughly the delay experienced. The cooling is a little bit slower, but we still need to anticipate the delay : the value of ΔT to take into account must be smaller : when the automaton opens the die after a decrease of ΔT, the real decrease of temperature is bigger.

In the case of the test die, the ΔT measured was 12°C. After trial and error adjusting ΔT, the working value is ΔT = 4.5°C.

2. Solutions - Hardware

2.1. Motion of the Cylinders

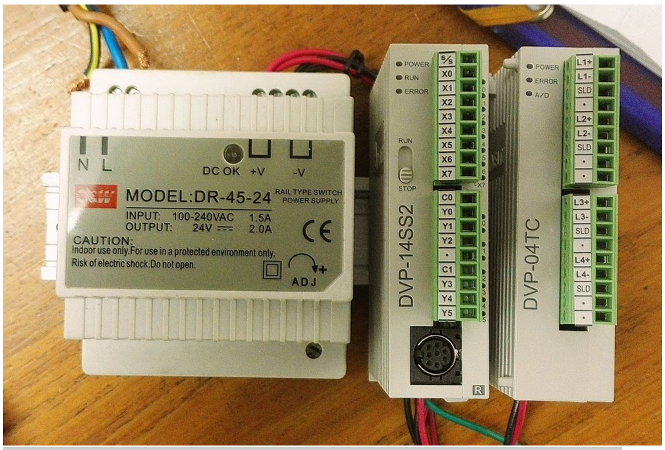
As said, the table was already in use before so the components for the basic moves (opening, closing) were installed :

- The two **fixed plates** 600mm apart, and the **moving plate** in between
- Two hydraulic **cylinders** (diameter 76mm) one above the other. They function as one single cylinder, since the hydraulic pipes divide the oil flow into both cylinders. It pushes and pulls the moving plate
- Two symmetrical command systems, one to close and one to open. They are working in 24V :
  - The **left button** lets current through the **left relay**, which lets current through the **left solenoid valve**. Linked to the back of the cylinders, it closes the table.
  - The **right button** lets current through the **right relay**, which lets current through the **right solenoid valve**. Linked to the front of the cylinders, it opens the table.
- The **transformer** supplying the 24V from the 230V power source
- The **emergency stop button** cutting the 230V from the transformer

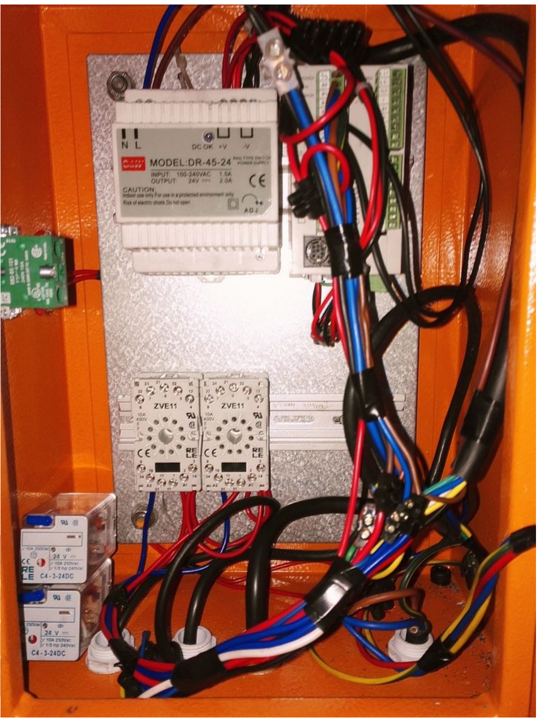
The relays can be considered as the simplest 1 Input / 1 Output devices :

- Input ON ⇒ Output ON
- Input OFF ⇒ Output OFF

In order to automate the table, the relays were the parts I had to replace. We replaced both relays by one single automaton : a Programmable Logic Controller (**PLC**), a device with 8 Inputs / 8 Outputs. Firstly, it took the two buttons as inputs (**X0** and **X1**) and the two valves as outputs (**Y0** and **Y1**), to function just like the relays. The difference is that the PLC (DVP14SS2-11R by Delta) is programmable, with its associated software (WPLSoft), in the Ladder Diagram language.


N L
DC OK +V -V
MODEL:DR-45-24
RAIL TYPE SWITCH POWER SUPPLY
INPUT: 100-240VAC 1.5A
OUTPUT: 24V 2.0A
CAUTION: Indoor use only.For use in a protected environment only. Risk of electric shock.Do not open.
ADJ
POWER
RUN
ERROR
X0 X1 X2 X3 X4 X5 X6 X7
RUN
STOP
C0 Y0 Y1 Y2 C1 Y3 Y4 Y5
DVP-14SS2
POWER
ERROR
A/D
L1+ L1- SLD L2+ L2- SLD L3+ L3- SLD L4+ L4- SLD
DVP-04TC

Transformer ⇧ , PLC ⇗ and Temperature module ⇧



Layout of the electrical box. ⇨

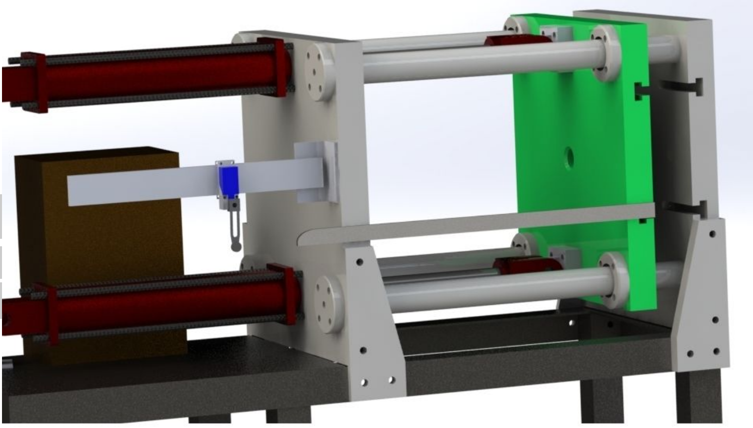
The relays (bottom left) are unplugged.

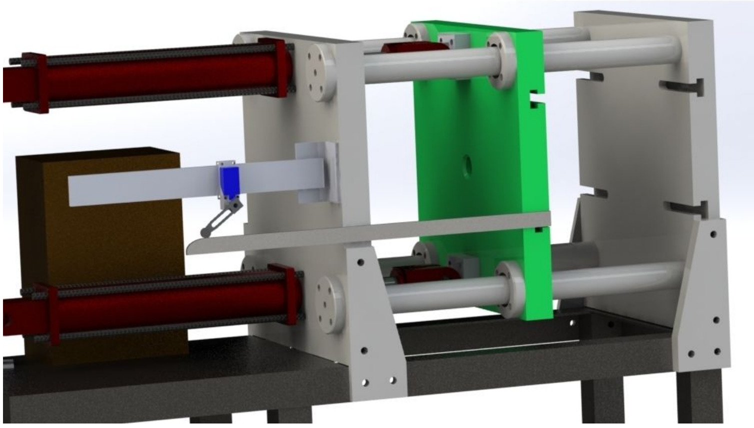
The relays and the PLC can be unplugged / replugged easily, it facilitates switching between both systems

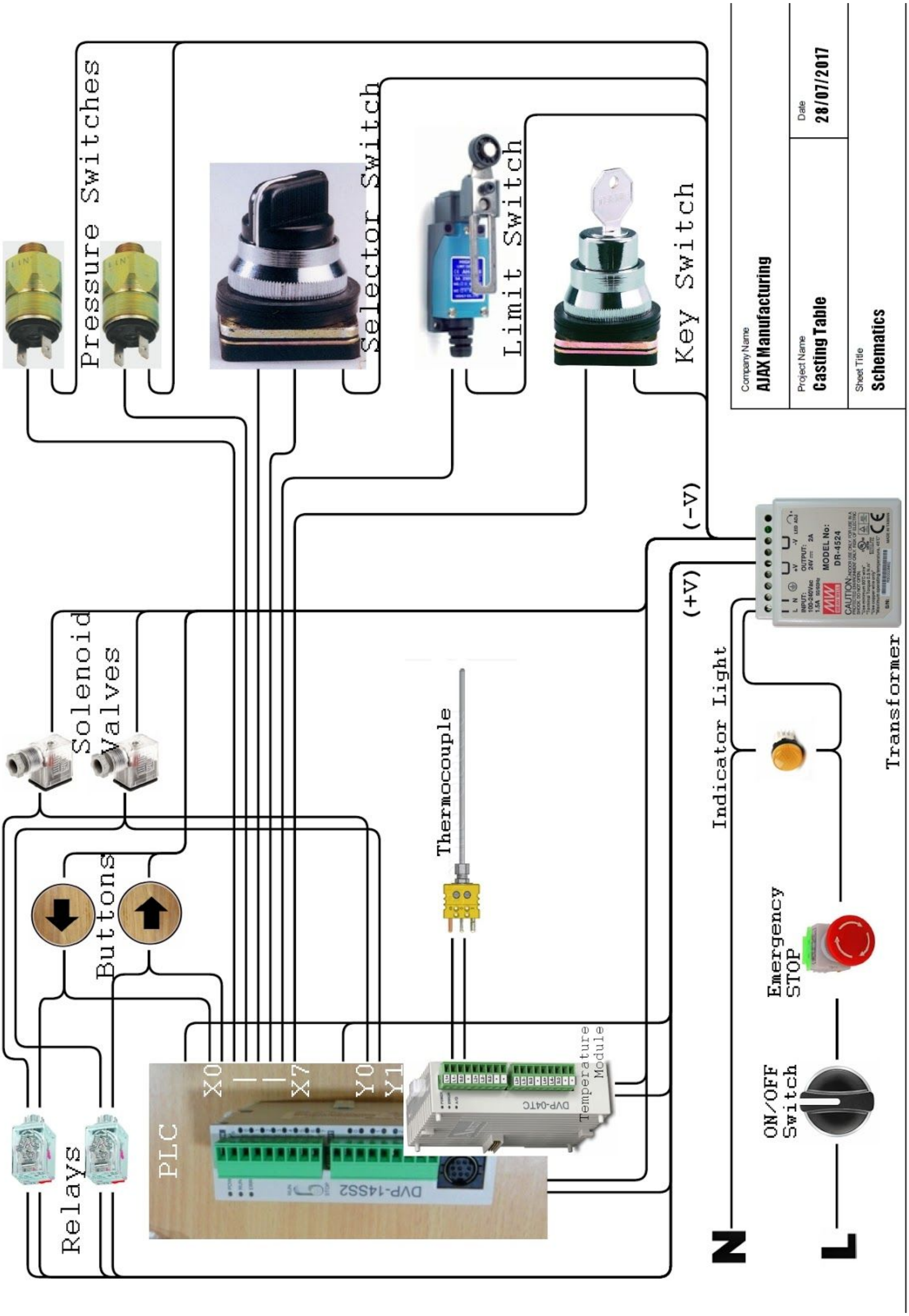
To operate the desired cycle (see 1.1. Study of the Cycle), I had to add to the system the following components, purchased from the known suppliers of Ajax :

- Two **pressure switches** (PMN-2C by HyControl) :
  - ➢ Like the valves, they are part of the hydraulic <u>and</u> the electronic systems. As the valves detect an electronic signal and open the oil flow, the pressure switches do the opposite. They detect if the pressure reaches some level (which can be set up), and close the electronic circuit to send a signal.
  - ➢ They are wired to the PLC as inputs (**X2** and **X3**), and are plugged to the edges of the cylinders : it informs the PLC when the cylinders reaches the end of its course.
- A **selector switch** (AS2-3P by ACDC Dynamics) :
  - ➢ This switch with 3 positions allows the operator to choose between the three modes (hand, semi-automatic, automatic). It wires two inputs (**X4** and **X5**) and can send three different signals :
    - X4=ON |X5=OFF : Hand mode
    - X4=OFF|X5=ON : Semi-automatic mode
    - X4=OFF|X5=OFF : Automatic mode
- A **limit switch** (AH8108 by ACDC Dynamics) :
  - ➢ It can detect a mechanical movement by physical contact, and close the electronic circuit to send a signal. It detects when the moving plate reaches some position, and sends a signal to the PLC (input **X6**)
  - ➢ With a system of welded rods screwed to the moving plate, it can be adjusted to fit the corresponding needs (trigger at 10% of the opening, or 50% of the closing, etc…)
- A **key switch** (AK-2PB by ACDC Dynamics) :
  - ➢ It functions like the selector switch but with only 2 positions, and requires a physical key to switch. It sends a signal (input **X7**) when the key is in the device, meaning that the user is administrator.

The limit switch (in blue). ⇨
It is adjustable to detect the position of the middle plate.







Company Name
AJAX Manufacturing

Project Name
Casting Table

Sheet Title
Schematics

Date
28/07/2017

Schematics of the system

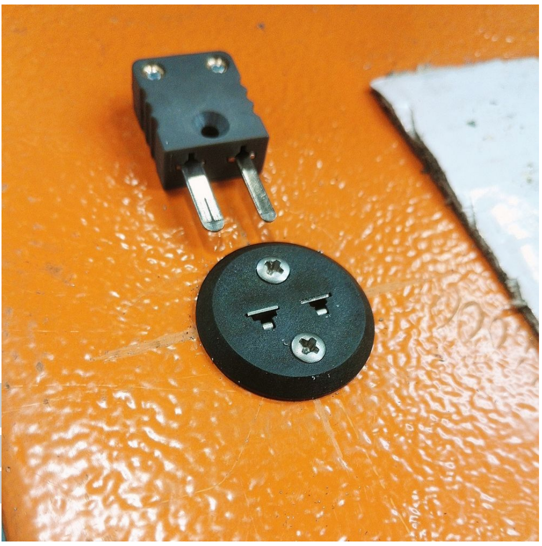
2.2. Temperature Sensors

Each die has its own specificities. The differences in material, thickness, geometry, and the different manners to build the sensor in the shell, cause different **thermal behaviours**. Therefore the sensor must be part of the die : it is built in it and is not removed, for the temperature data to be consistent for the same die, from one casting session to another.

Ajax was already ordering thermocouples on a regular basis, and the supplier (Unitemp) offers to make customized thermocouples. The ones ordered for this project are **J-type thermocouples**, with a linear range from **0°C to 750°C**, and a teflon tubing offering a physical resistance up to 800°C.

In the die, I drilled and tapered a hole that fits perfectly to the shape and the threading of the thermocouple. We found out that the part between the sensing tip and the teflon tubing was exposed to the high temperature flames of the blowtorch, so a metal protection around this part happened to be necessary.

Every time the die is changed, the thermocouple needs to be changed. But it also needs to be plugged to the PLC which is inside the electrical box. That is why I also ordered and set up pairs of plugs (male & female) ; now the thermocouple can be plugged / unplugged easily, without having to open the electrical box.



The female plug is built in the box ⇨

The male plug will be wired to the edge of the thermocouple.

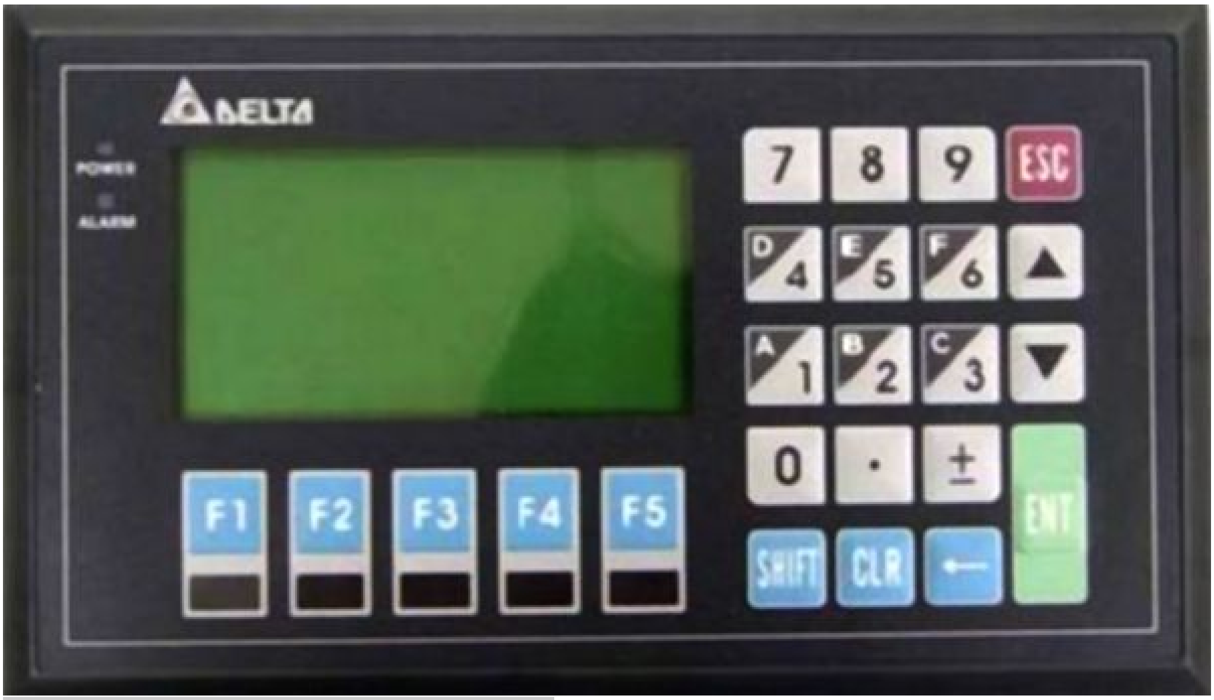
The PLC can only receive binary inputs, but the manufacturer proposes some supplementary modules for different tasks. The **Analog Temperature Module** has thermocouple inputs, 2 wires for each input. It is plugged to the PLC.

It sends the data continuously : one temperature value of with a **precision of 0.1 °C**, updated every 0.1s.

2.3. Human-Machine Interface (**HMI**)

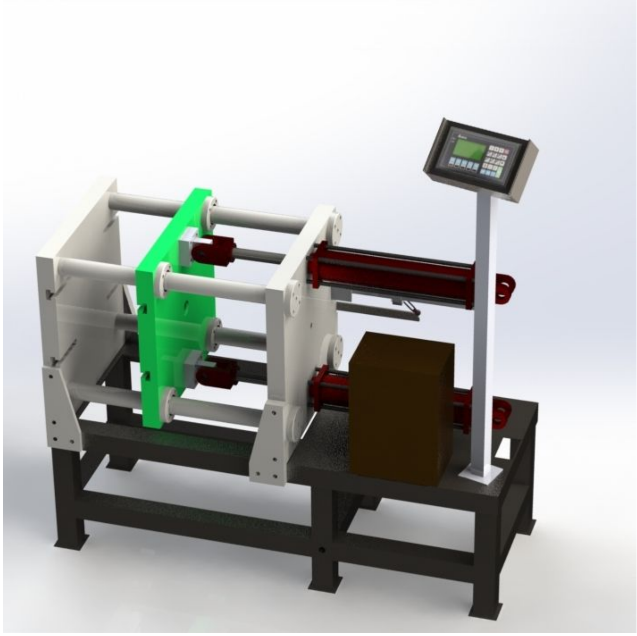
We cannot let the PLC operate the table autonomously without interacting with the user ; the implementation of a screen was necessary. From the different HMI solutions possible to work with the PLC, I chose a **text panel** (TP08G-BT2 by Delta). It is programmable, with its associated software (TPEditor) :

- System to user : The screen, **240x128** pixels in black or white. It is mainly used to display the **instructions**, corresponding to the current step of the cycle, but also
  - the I.D. of the die
  - the mode currently in use
  - the live temperature from the thermocouple
- User to system : The buttons, to let the operator enter the die I.D. and its specs.



⇧ *The face of the text panel : screen + buttons*

The screen now needed to be built on the table. Employees from the maintenance taught me to weld metal plates together. It allowed me to make a case for the screen; with a plexiglas protection and a leg to screw the device to the table and pass the wires into the electrical box.



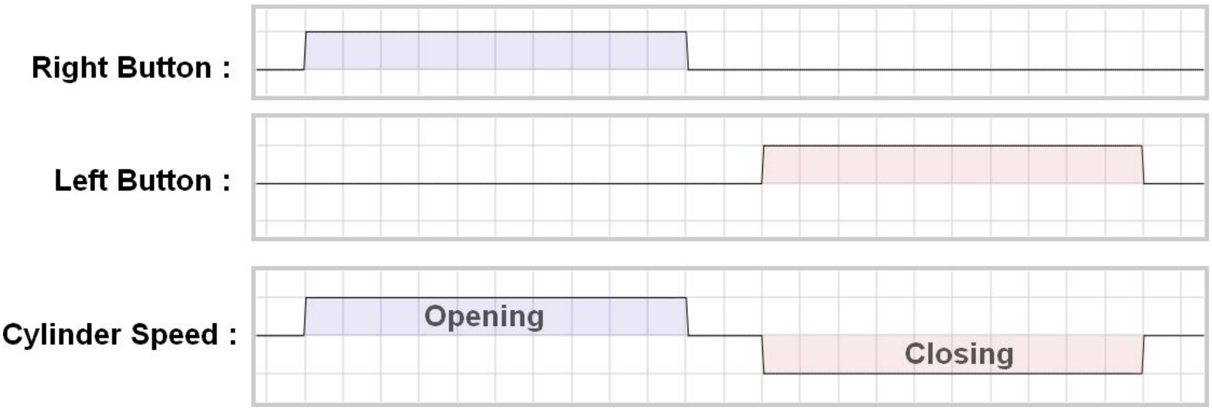
⇖ *Implementation of the text panel*

3. Solutions - Software

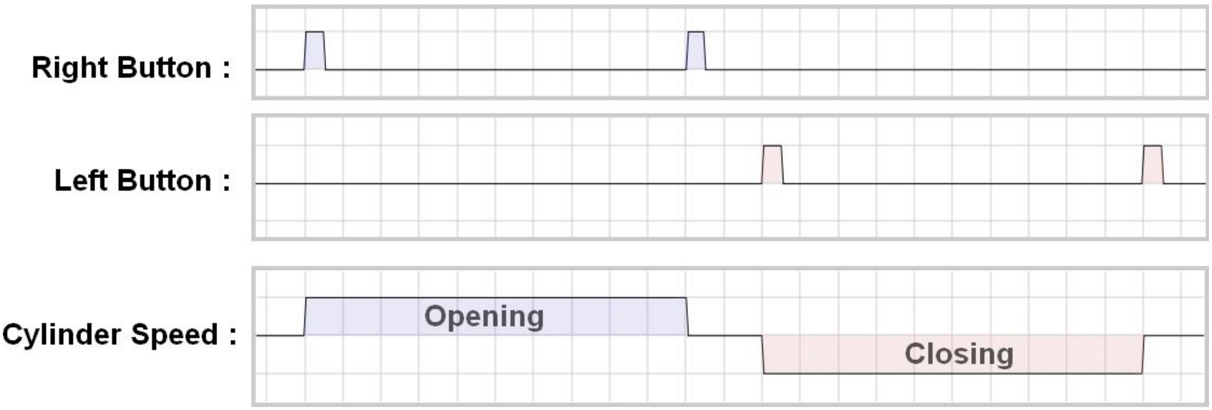
3.1. PLC General Program

To avoid any safety problems, the operator must be able to take full control of the machine at any moment. This is the purpose of the **selector switch (3 positions)** allowing the operator to set the table in 3 different modes :

a) The **'hand mode'**, which was basically the already existing mode. In hand mode, the operator must keep the buttons pressed to move the plate, as shown below:

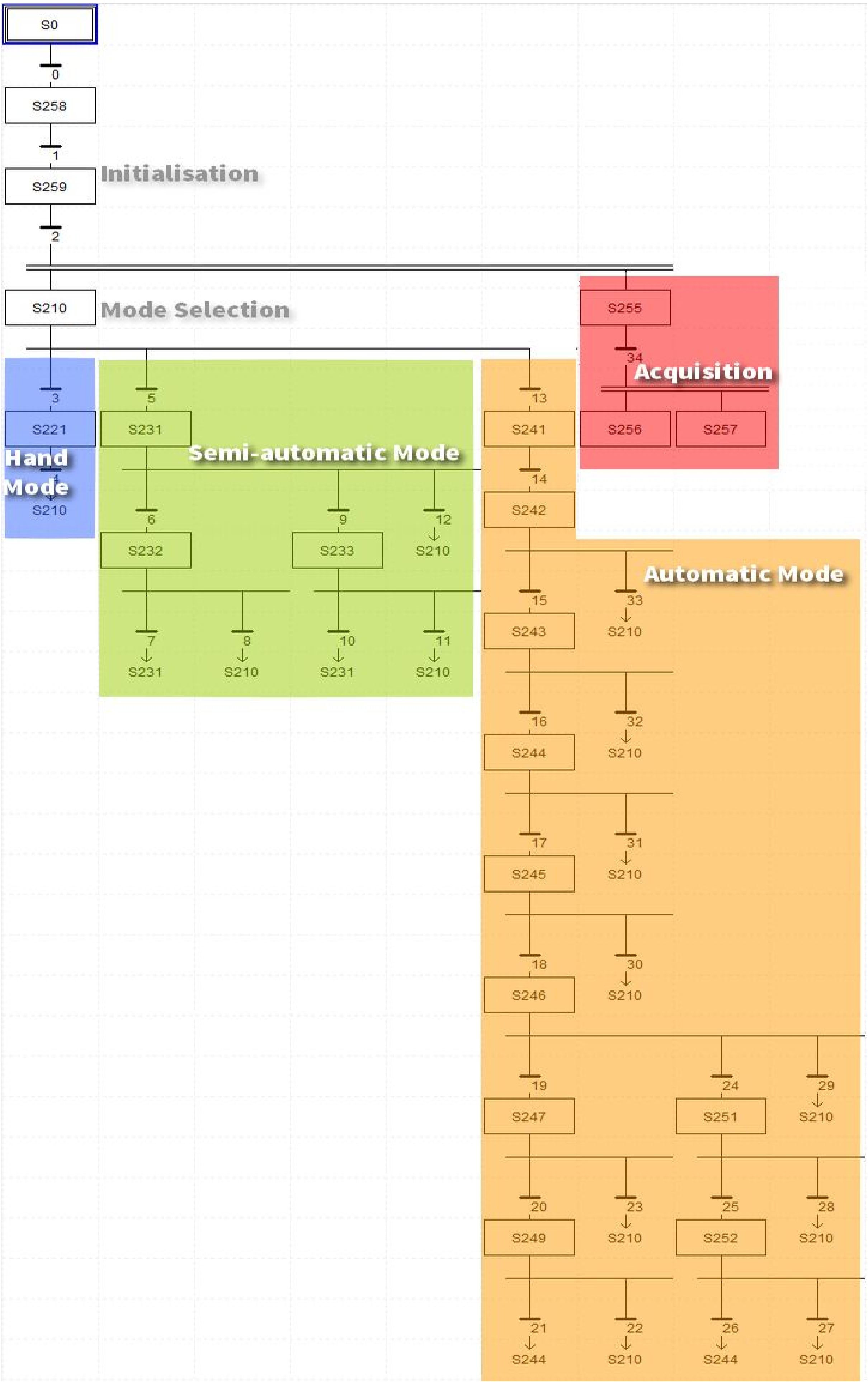


b) The **'semi-automatic mode'** is just more convenient for the worker. In semi-automatic mode, the buttons have to be pressed once to move the plate, and once again to stop the movement, as shown below:



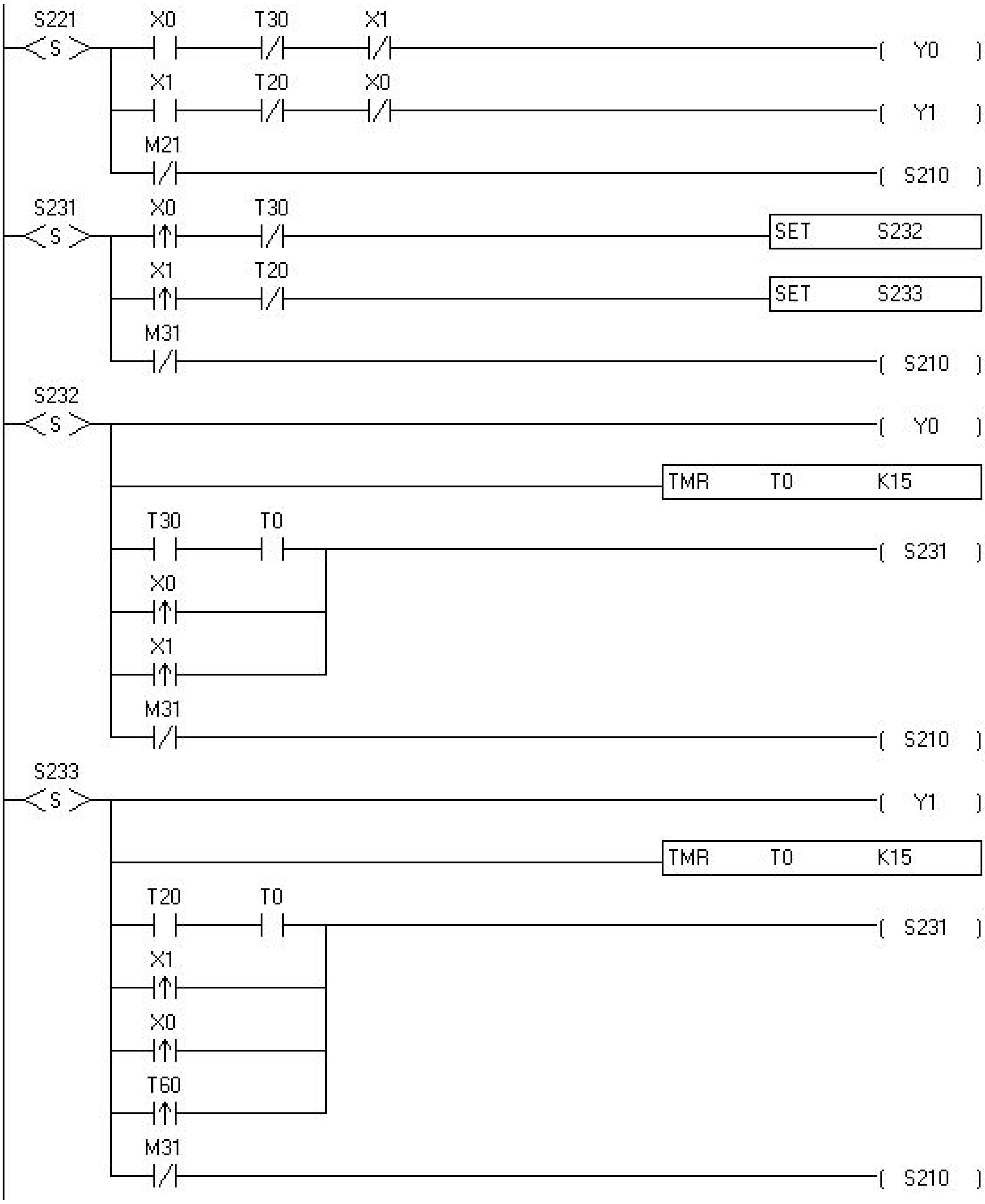
c) The **'automatic mode'** was the main goal, where the PLC works along the cycle. There are a few key moments in the cycle when the operator has to press a button, to let the machine know that the corresponding task is actually done (see : 1.1. Study of the Cycle). Apart from that, the number of human interventions on the buttons is kept to the minimum. This is after all the purpose of any automatic mode.

Here is the global functioning of the program that I implemented within the PLC, represented in Sequential Function Chart (**SFC**) :



⇧ *The architecture of the program. Made in the SFC mode of WPLSoft.*

Each cell and each link of the SFC Diagram are composed of **Ladder** Diagrams.
I put the detailed program and its comments at the end of the report (**Appendix A**), but as an example here is the part corresponding to the **Hand** and **Semi-Auto** modes :



Hand mode. X are the buttons, T are the pressure switches, Y are the valves

Quits when we change the mode

*Semi-Auto mode. It works the same way, but takes only pulses from the buttons. Instead of setting Y0 or Y1, it sets **S232** or **S233** which will keep the action going*

S232 closes the die, until either a button is pressed, or T30 (pressure switch) is triggered. T30 is disabled for the first 1.5s, in order to let the cylinders run, to unclog the pressure switch

*Same thing to open the die, but this time we also take into account **T60 (limit switch)** as well*

Each die is attributed 4 data : its I.D., and the 3 temperature setpoints. In the specs provided by the supplier of the PLC, we see that there is a range of 1920 ***Latched*** Data *Register* (from $\mathbf{D}_{2000}$ to $\mathbf{D}_{3919}$), meaning that it will remain in memory even when the PLC is shut down. There we can store 1920/4 = 480 different dies. I found it convenient to use it as below :
From N=1 to N=480 :

- $\mathbf{D}_{1999+N}$ is the **I.D.** of the die
- $\mathbf{D}_{2479+N}$ is the starting temperature $\mathbf{T}_0$
- $\mathbf{D}_{2959+N}$ is the target temperature $\mathbf{T}_1$
- $\mathbf{D}_{3439+N}$ is the delta **ΔT**

And now we fully use the range from $\mathbf{D}_{2000}$ to $\mathbf{D}_{3919}$.

3.2. Temperature Acquisition

For the research at the beginning, I needed to study the thermal behaviour of the die during the casting process. In order to obtain the graph of the real temperature evolution (see 1.2. Study of the Temperature), I implemented in the PLC a **Ladder program** completely different and independant from the one controlling the cylinders.

This time, the goal was to tell the PLC how and when to send the measured data to a computer. Technically it sends **ASCII data** in an RS232 cable to a computer, then a **VBA script** will process the data into an **excel file**. Both the Ladder and the VBA are detailed at the end (**Appendix B**) with their comments, but here is the basic functioning :

There is only 128 different ASCII bytes possible : for a precision of 0.1°C, the range of temperature is only 12.8°C. However we can divide the information in **2 bytes**, so that we can send 128^2 = 16384 different values ; the range is now 1638.4°C which is enough.

Once the temperature T is stored in the PLC, we need to split it in two bytes (a single ASCII byte takes a value **between 0 and 127**). We could divide T by 128, then send the result D and the remainder R. The receiving program will just have to reverse the operation : ***T = D x 128 + R***, with D < 128 and R < 128. That way we send data packets (D, R). We need to separate these packets to let the receiver know which one is D, which one is R.

The solution I adopted is to declare the integer 127 as the “stop byte”. We send it between two data packets : (D, R), 127, (D, R), 127, etc… But now D and R cannot equal 127, otherwise it would confuse the receiver ; so instead of dividing T by 128, we divide by 127 so that ***T = D x 127 + R***, with D < 127 and R < 127.

Now the integer 127 is free to use as an “information byte” (In that case it is a “Stop Byte”). We also want to free the integer 126 in case we want it to be an “Error Byte” ; so we repeat that process. **T = D x 126 + R**, with D < 126 and R < 126.

Then the PLC sends continuously (D, R), 127, (D, R), 127, etc…
(see Appendix B for more details)

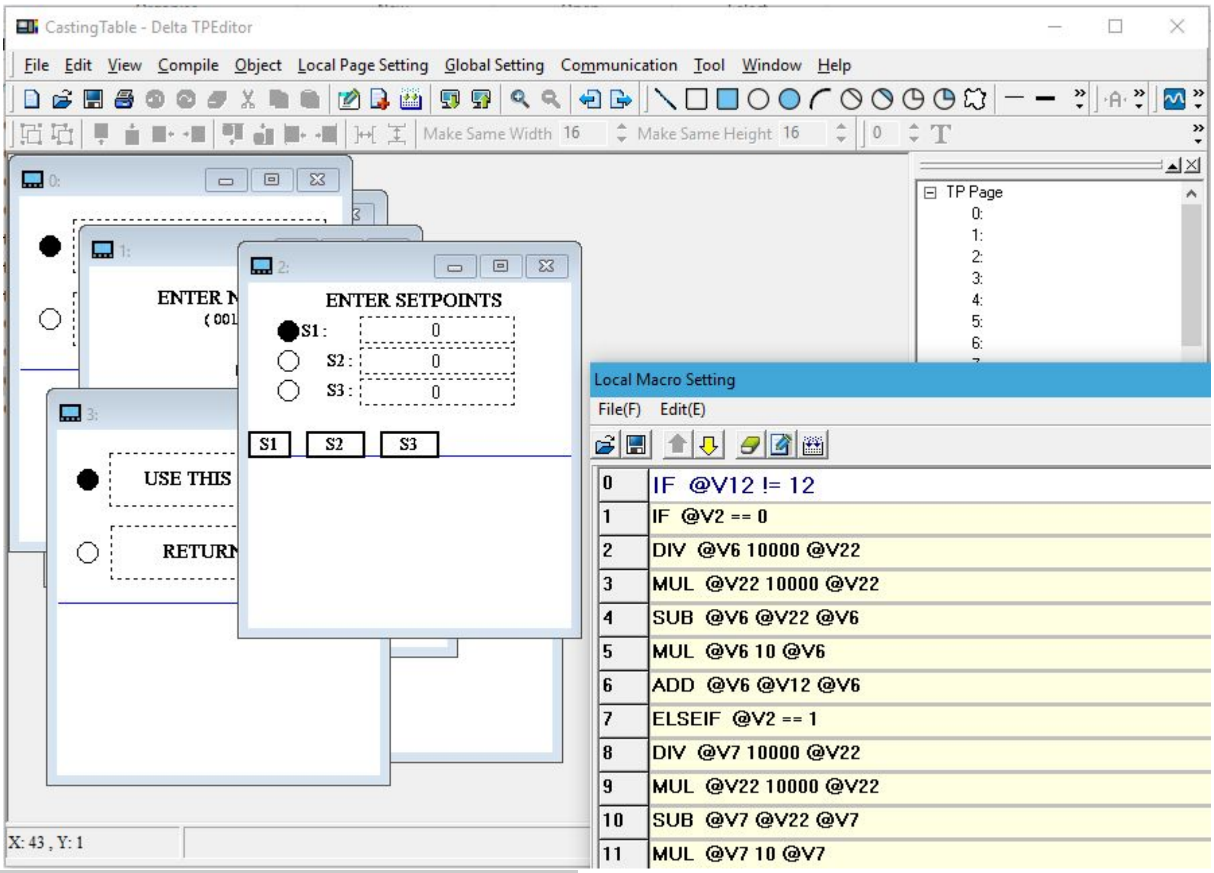
3.3. Graphical Interface

The text panel set up on the table (see 2.3. Human-Machine Interface) came with the software TPEditor to program it. It allowed me to design pages, and link them with three sorts of features :

- A ***Page Jump*** takes a condition and a target page, it does the jump at the very moment the condition is True.
- A ***Function Key*** associates one of the physical buttons to an action. It can be a page jump, a data modification, a relay, etc…
- A ***Macro*** can do everything ; however the software is not perfect and I found it very unstable with macros. It works on the moment, but after closing and reopening the file, the macros randomly switched the page they were attributed to.

Each of these 3 features can be attributed to a specific page and will only be active when the page is displayed (**Local Page Setting**), or defined as a global feature, always active (**Global Setting**).

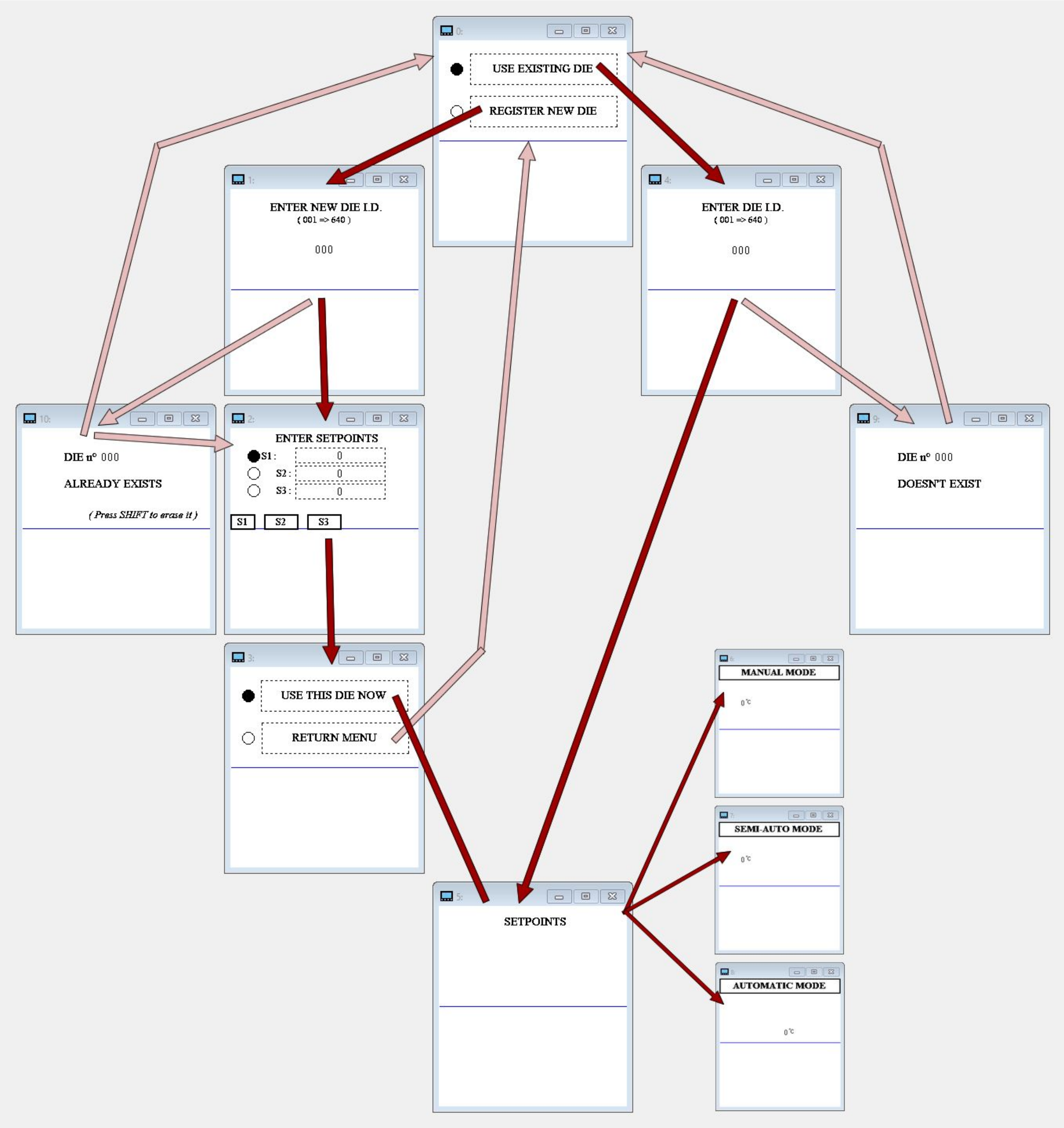
The goal, before displaying the instructions and the temperature, was to define what die is going to be used, to inform the PLC of the different temperature values that are specific to the die (see 1.2. Study of the Temperature). It can lead either to register a new die with a new I.D., or enter the I.D. of an already registered die.



```
IF @V12 != 12
IF @V2 == 0
DIV @V6 10000 @V22
MUL @V22 10000 @V22
SUB @V6 @V22 @V6
MUL @V6 10 @V6
ADD @V6 @V12 @V6
ELSEIF @V2 == 1
DIV @V7 10000 @V22
MUL @V22 10000 @V22
SUB @V7 @V22 @V7
MUL @V7 10 @V7
```

⇧ *Software TPEditor used to program the screen*

On the next page, the layout of the interface :



⇧ The architecture of the graphical interface

Conclusion

The first days of work, I began to grasp the entirety of the assignment. The last week, I had to prepare the documentation for the next person to pick up the work. In both situations and all of those in between, I discovered that diverses qualities were needed, outside of the technical skills required to complete the assignment.

Regarding the organization side, I found many elements that I had already observed in the context of a company : how every purchase is registered and followed, how every task is labelled under a job number, every quality problem is traced back to its root, etc…

However this time I was not just observing this, but was in the middle of this system. It makes a huge difference, since it forced me to comply to this functioning. It also evolved during the three months of the internship : safety signs and clear instructions were installed next to the sensitive workplaces, for instance the powerful machines and the hot furnaces.

An intern in human resources has set up easier ways for the workers to be heard. Sometimes people become really annoyed with hierarchic relations, and a department destined to handle these situations is not useless at all.

The technical aspect was really satisfying because of the learning rate I had to maintain. I had at my disposal a lot of industrial machines to cut, grind, weld, grind, etc… The way from designing a piece to finishing it and implementing it into the casting table is now within my reach, and in an autonomous manner.

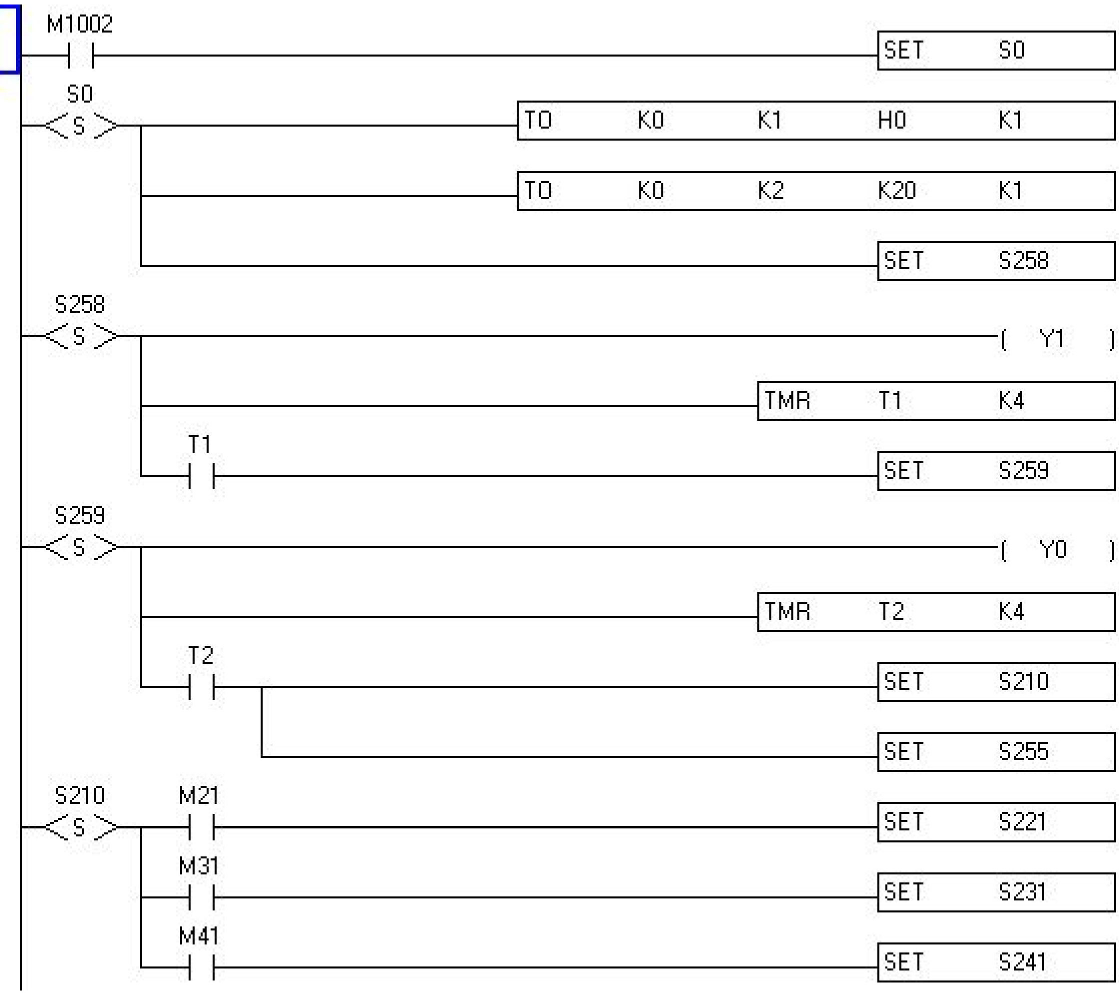
In the fields of mechanics and electronics, I found it essential to always have a **global perspective** of what I was doing. Dividing a single problem in smaller accessible problems is useful, but not if we forget to think about the whole system. This is what I realized in this project in which completely various aspects were mixed together.

The topic of this internship really was a continuity of what interested me in my years of engineering school : working on solutions to create or improve a mechatronic system. In addition to that, during the internship I had been accepted in the research master that I wanted, which is also orientated towards robotics. Knowing that all I was doing was preparing me to the next stage of my studies, it really boosted my motivation in the work.

Appendix A

Detail of the final ladder program, for the casting table :

A. Initialisation



Sets thermocouple type as J-type

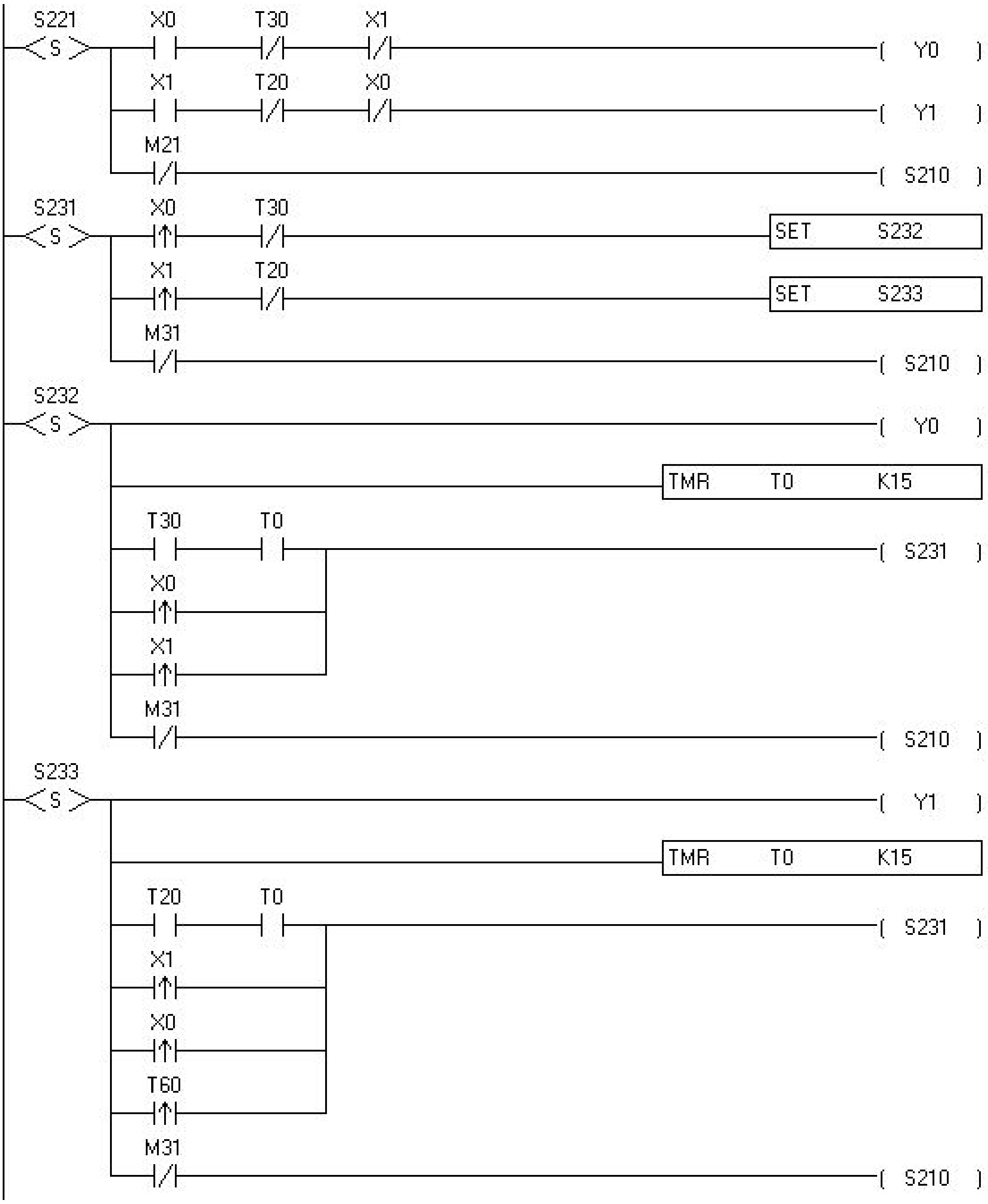
Sets measurement duration as 2s

Opens the die for 0.4 seconds, and closes the die for 0.4 seconds.
It shows that the table is switched on, and unclogs the pressure switches

*Sets the steps **S210 (selects the mode)** and **S255 (acquire variables)** simultaneously.*

M21 = Hand mode
M31 = Semi-Auto mode
M41 = Auto mode
*They are switched on/off in **S257***

B. Modes **Hand** and **Semi-Auto**



Hand mode. X0 and X1 are the buttons T30 and T20 are the pressure switches

Quits when we change the mode

*Semi-Auto mode. It works the same way, but takes only pulses from the buttons. It doesn't set Y0 or Y1, but **S232** or **S233** which will keep the action going*

.
.
.

*S232 closes the die, until either a button is pressed, or **T30 (pressure switch)** is triggered. T30 is disabled for the first 1.5s, in order to let the cylinders run to unclog the pressure switch*

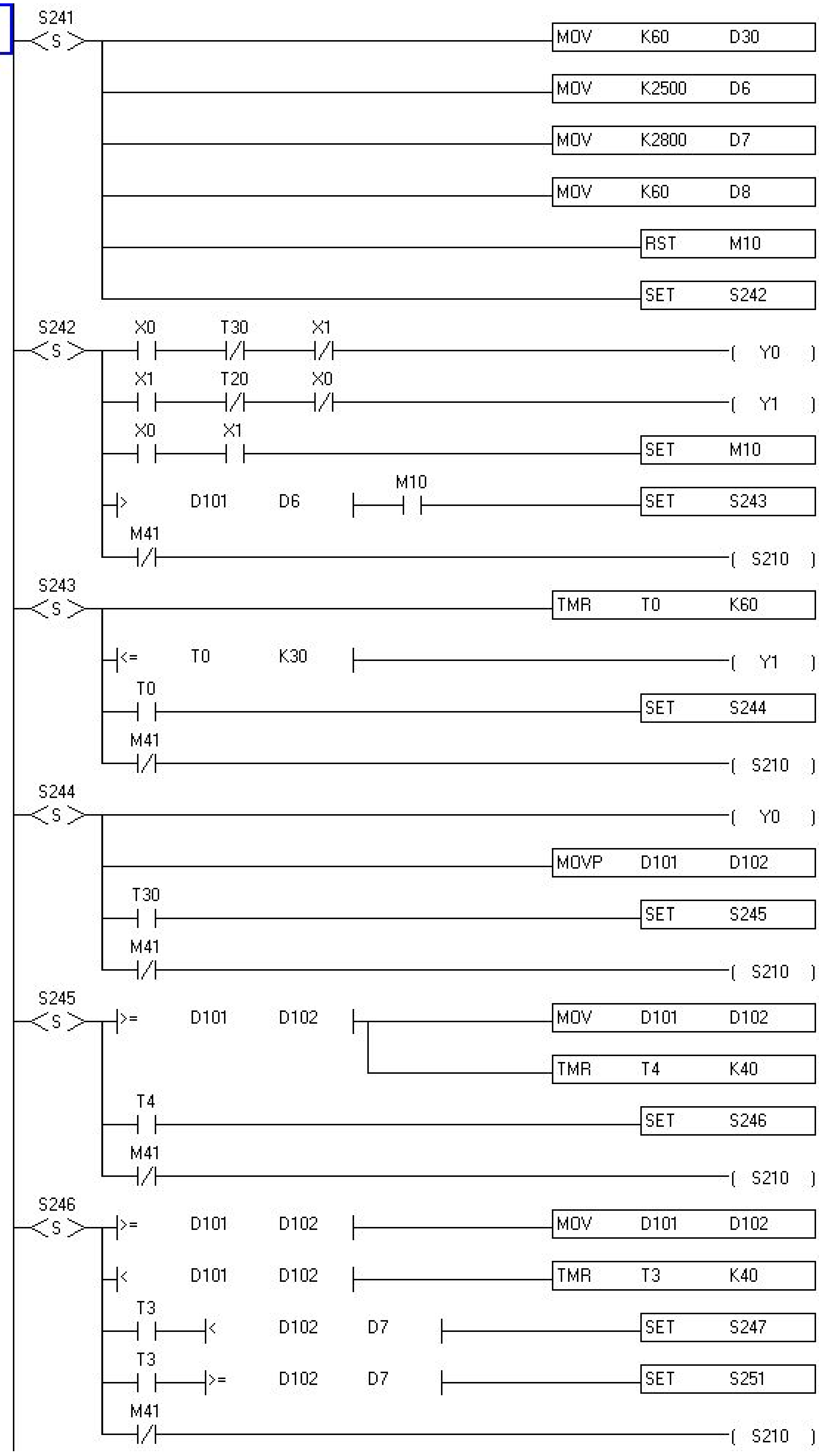
.
.
.

.
.
.
.

*Same thing to open the die, but this time we take into account **T60 (limit switch)** as well*

.
.
.
.

C. Mode Automatic



D6, ***D7*** *and* ***D8*** *are the 3 key values, that are specific to each die. Here it is defined manually (**250°C**, **280°C**, **6°C**) but the goal will be to import it from what the user enters on the screen*

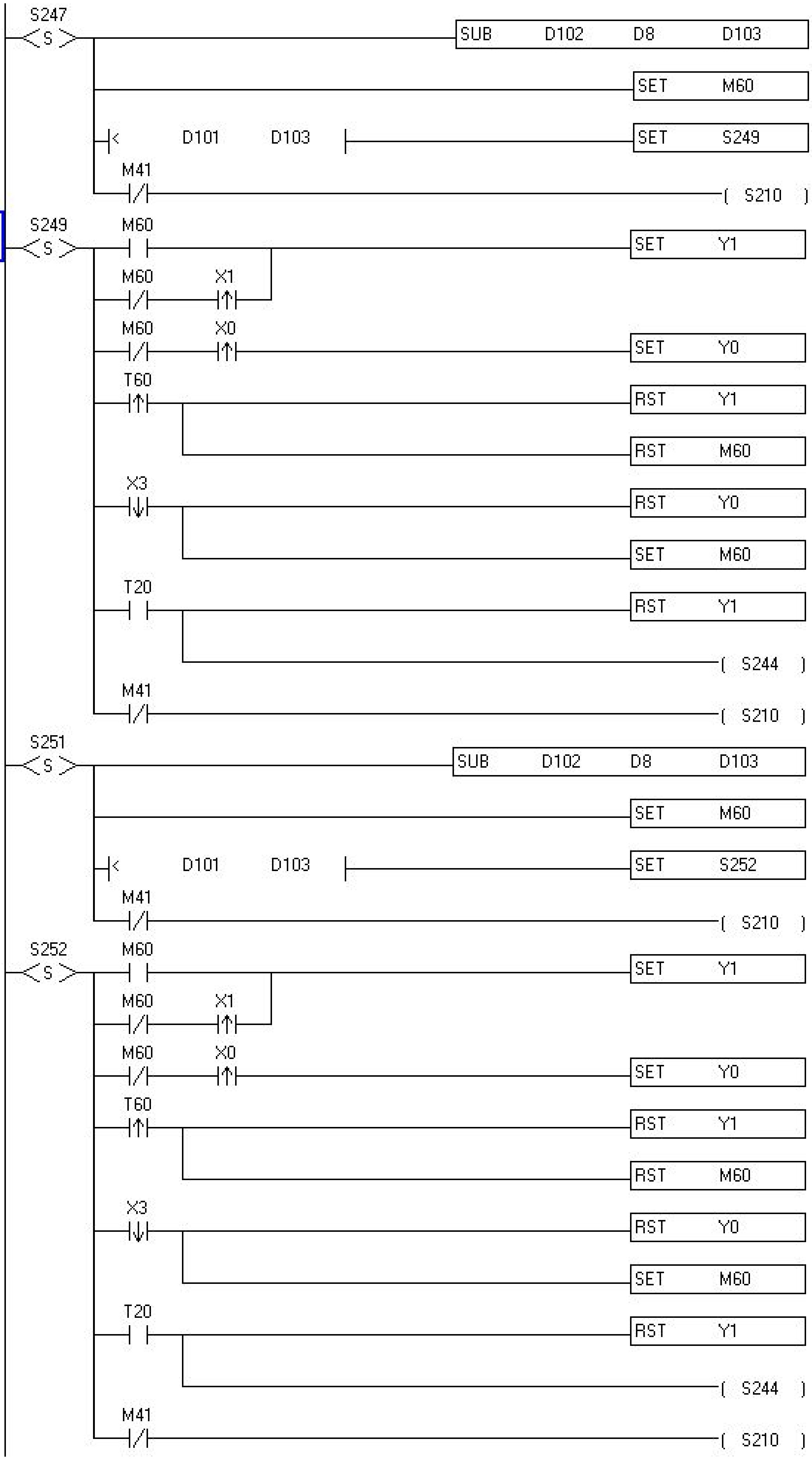
S242 *is almost a copy of the **hand mode** : we begin by heating the die (and spray the coating), so we have to control the cylinders manually.*
*The next step is enabled when **$T>T_0$**, but the user has to press both buttons to say "ready to start the castings"*

.
*For 3 seconds the die opens, to **warn** of the imminent closing. Then we have 3 seconds of pause*
.

*The die closes until the **pressure switch T30** is triggered.*
*We also put the temperature D101 in the data **D102** which will be the **maximum temperature***

.
*D101 fluctuates (it is **automatically updated in S256**) and D102 keeps the maximum temperature*
.

*We **pour the aluminium**. D101 and D102 work the same way as above, but we also wait for the temperature to cool down for at least 4s : then we know the temperature peak is reached.*
*The **peak value T_{max}** is in D102, and is compared to **T_1 (the 2nd key value)**. S247=Scrap piece, S251=Good piece*



.

Now we need the aluminium to cool down to ***$T{=}T_{max}$-ΔT*** *(D103=D102-D8). ΔT is the* ***3_{rd} key value****, stored in D8. We also* ***set M60*** *for the next step*

.

Now is the delicate part : ***taking the piece out of the die****. Most of the time we just have to open the die, but sometimes we have to* ***slightly open / close*** */ open / close the die repeatedly to make the piece break off correctly.*

Following the sequence :

- M60 is already set, so ***Y1 (opening) is set*** *until the limit switch T60 triggers, and* ***resets Y1*** *and M60, so the movement stops.*

*- Then the PLC waits for a button (****X0*** *or* ***X1****).*

- If the piece sticks, press ***X0 (close)*** *so* ***Y0 is set****. The die closes completely until the pressure switch X3 triggers. It* ***resets Y0****, and sets M60 so we start over : the die opens and stops to wait for a button, and we press either X1 or X0.*

- If all is good, we press ***X1 (open)*** *so* ***Y1 is set again*** *and the die opens. It opens to the maximum (pressure switch T20) and the ejector pins allow us to take the piece out of the die.*

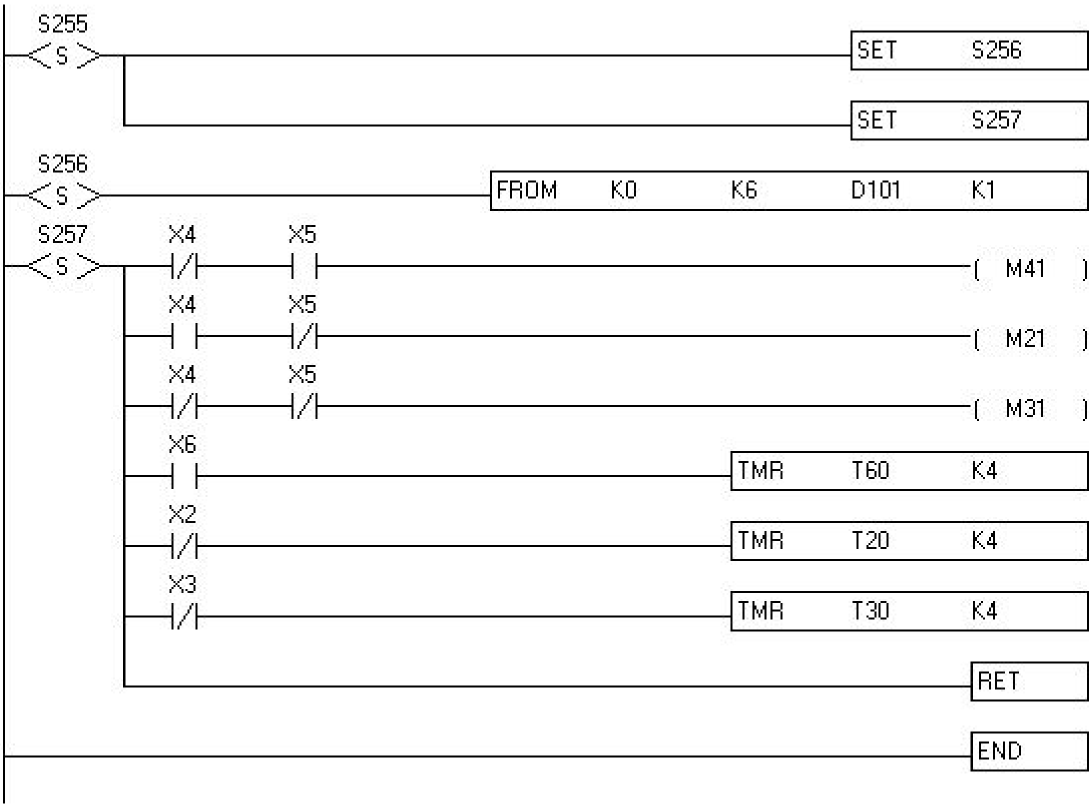
We then go back to S244 to cast the next piece

.

S247 / S249 was in the case of a scrap piece. ***S251 / S252 is exactly the same*** *but in the case of a good piece. I separated two cases to treat them the same way : although it is useless for now, it will make the future programming easier if we want to change what happens in only one of the cases*

.

D. Acquisition



*S255 sets **S256 (Temperature)** and **S257 (Inputs)**. They are running continuously and never stop*

*Temperature(**x 0.1°C**) is stored in D101*

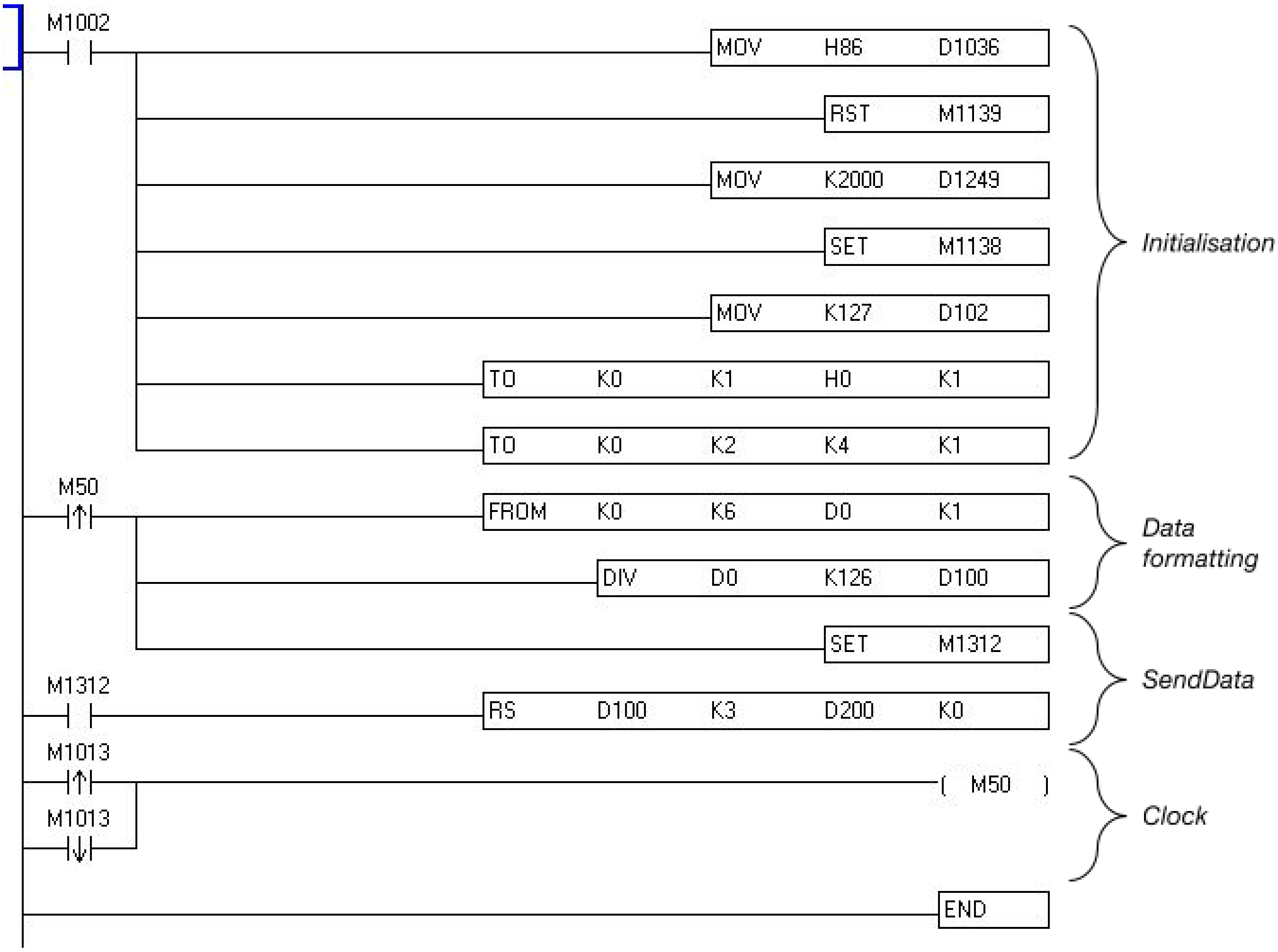
The mode is controlled by X4 and X5 coming from the selector switch
M21 = Hand mode
M31 = Semi-Auto mode
M41 = Auto mode

X2**, **X3** (pressure switches) and **X6** (limit switch) are sometimes triggered for a fraction of a second. We want to ignore these interferences so we create **T20**, **T30** and **T60** which will trigger only when their respective input is active for more than **0.4 second

***RET** & **END** are imperative at the end*

Appendix B

Detail of the Ladder program to implement in the PLC to do live temperature acquisition with excel (works with the VBA file underneath). It tells the PLC how and when to send the data. It sends ASCII data in an RS232 cable to a computer (and then the VBA of the excel file will process the data). There is only 128 different ASCII bytes, so for a precision of 0.1°C the range of temperature is only 12.8°C. We divide the information in two bytes, so that we can send 128^2 = 16384 different values ; the range is now 1638.4°C which is more than enough.



- Initialisation :
  - **MOV H86 D1036** : Sets up communication protocol as 9600, 7, E, 1
  - **RST M1139** : Sets up ASCII mode
  - **MOV K2000 D1249** : Communication time-out is 2s
  - **SET M1138** : Ready for data sending
  - **MOV K127 D102** : D102 = 127 (useful later)
  - **TO K0 K1 H0 K1** : Sets up thermocouple type as J-Type
  - **TO K0 K2 K4 K1** : Sets up measurement duration as 400 ms

- Data formatting :
  - **FROM K0 K6 D0 K1** : Stores the temperature in D0

  Then we need to split D0 in two bytes : D100 and D101. We could divide D0 by 128, put the result in D100 and the remainder in D101. The receiving program will just have to reverse the operation : T = D100 x 128 + D101, with **D100 < 128** and **D101 < 128**. That way we send data packets(D100, D101). We need to separate these packets to let the receiver know which one is D100, which one is D101.

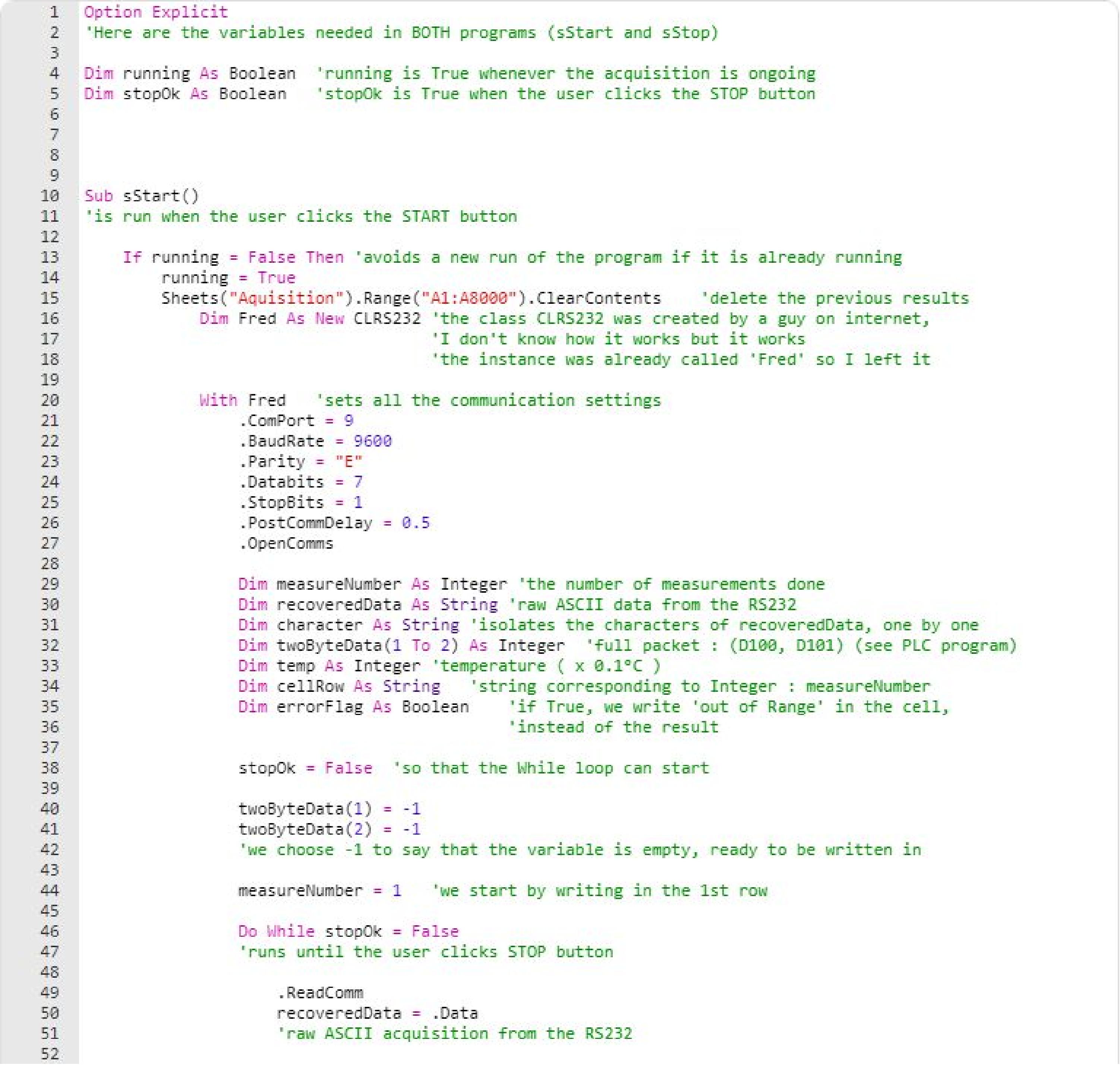
  The solution adopted is to declare the integer 127 as the "stop byte", that we send between two data packets : (D100, D101), 127, (D100, D101),127, etc… But now D100 and D101 cannot equal 127, otherwise it would confuse the receiver ; so instead of doing *D0 = D100 x 128 + D101*, we divide D0 by 127 so that T = D100 x 127 + D101, with **D100 < 127** and **D101 < 127**.

  Now the integer 127 is free to use as an "information byte", in that case it is a "Stop Byte". We also want to free the integer 126 in case we want it to be an "Error Byte" ; so we repeat that process. **T = D100 x 126 + D101**.
  - **DIV D0 K126 D100** : Divides D0 by 126, puts dividend in D100 and remainder in D101 automatically.

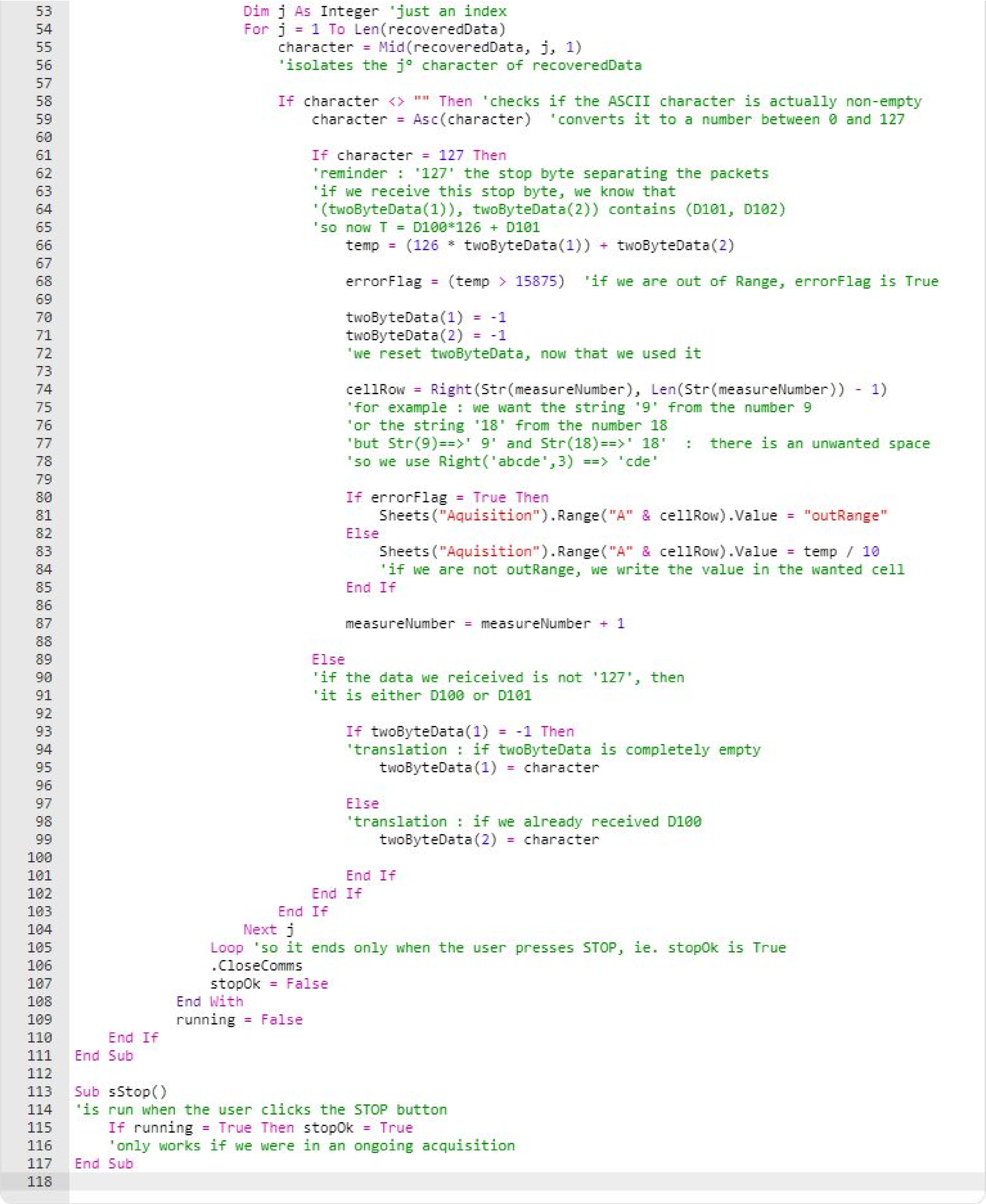
  Now we can send only 126^2 = 15876 different values, it still makes a range of 1587.6°C, more than enough.

- Send data
  - **SET M1312** : Triggers data sending
  - **RS D100 K3 D200 K0** : Sends 3 data : D100, D101, D102 (D102 = 127) and receives 0 data.
- Clock:
  - M1013 is a built-in clock, which is ON for 0.5s then OFF for 0.5s.
    By taking each rising edge and falling edge, M50 is a pulse every 0.5s.

Here is the VBA file to write in excel to do live temperature acquisition (in excel 2007)



```vb
Option Explicit
'Here are the variables needed in BOTH programs (sStart and sStop)

Dim running As Boolean  'running is True whenever the acquisition is ongoing
Dim stopOk As Boolean   'stopOk is True when the user clicks the STOP button




Sub sStart()
'is run when the user clicks the START button

    If running = False Then 'avoids a new run of the program if it is already running
        running = True
        Sheets("Aquisition").Range("A1:A8000").ClearContents    'delete the previous results
            Dim Fred As New CLRS232 'the class CLRS232 was created by a guy on internet,
                                    'I don't know how it works but it works
                                    'the instance was already called 'Fred' so I left it

            With Fred   'sets all the communication settings
                .ComPort = 9
                .BaudRate = 9600
                .Parity = "E"
                .Databits = 7
                .StopBits = 1
                .PostCommDelay = 0.5
                .OpenComms

                Dim measureNumber As Integer 'the number of measurements done
                Dim recoveredData As String 'raw ASCII data from the RS232
                Dim character As String 'isolates the characters of recoveredData, one by one
                Dim twoByteData(1 To 2) As Integer  'full packet : (D100, D101) (see PLC program)
                Dim temp As Integer 'temperature ( x 0.1°C )
                Dim cellRow As String   'string corresponding to Integer : measureNumber
                Dim errorFlag As Boolean    'if True, we write 'out of Range' in the cell,
                                            'instead of the result

                stopOk = False  'so that the While loop can start

                twoByteData(1) = -1
                twoByteData(2) = -1
                'we choose -1 to say that the variable is empty, ready to be written in

                measureNumber = 1   'we start by writing in the 1st row

                Do While stopOk = False
                'runs until the user clicks STOP button

                    .ReadComm
                    recoveredData = .Data
                    'raw ASCII acquisition from the RS232

```



```
                    Dim j As Integer 'just an index
                    For j = 1 To Len(recoveredData)
                        character = Mid(recoveredData, j, 1)
                        'isolates the j° character of recoveredData

                        If character <> "" Then 'checks if the ASCII character is actually non-empty
                            character = Asc(character)  'converts it to a number between 0 and 127

                            If character = 127 Then
                            'reminder : '127' the stop byte separating the packets
                            'if we receive this stop byte, we know that
                            '(twoByteData(1)), twoByteData(2)) contains (D101, D102)
                            'so now T = D100*126 + D101
                                temp = (126 * twoByteData(1)) + twoByteData(2)

                                errorFlag = (temp > 15875)  'if we are out of Range, errorFlag is True

                                twoByteData(1) = -1
                                twoByteData(2) = -1
                                'we reset twoByteData, now that we used it

                                cellRow = Right(Str(measureNumber), Len(Str(measureNumber)) - 1)
                                'for example : we want the string '9' from the number 9
                                'or the string '18' from the number 18
                                'but Str(9)==>' 9' and Str(18)==>' 18'   :  there is an unwanted space
                                'so we use Right('abcde',3) ==> 'cde'

                                If errorFlag = True Then
                                    Sheets("Aquisition").Range("A" & cellRow).Value = "outRange"
                                Else
                                    Sheets("Aquisition").Range("A" & cellRow).Value = temp / 10
                                    'if we are not outRange, we write the value in the wanted cell
                                End If

                                measureNumber = measureNumber + 1

                            Else
                            'if the data we reiceived is not '127', then
                            'it is either D100 or D101

                                If twoByteData(1) = -1 Then
                                'translation : if twoByteData is completely empty
                                    twoByteData(1) = character

                                Else
                                'translation : if we already received D100
                                    twoByteData(2) = character

                                End If
                            End If
                        End If
                    Next j
                Loop 'so it ends only when the user presses STOP, ie. stopOk is True
                .CloseComms
                stopOk = False
            End With
            running = False
    End If
End Sub

Sub sStop()
'is run when the user clicks the STOP button
    If running = True Then stopOk = True
    'only works if we were in an ongoing acquisition
End Sub

```

Synthèse en français

L'entreprise : AJAX Manufacturing et ses locaux

AJAX Manufacturing est une fonderie en 1951 au Cap, en Afrique du Sud. Elle débuta par le **moulage de pièces en aluminium**, toujours son activité principale aujourd'hui. Cependant Ajax a maintenant élargi ses activités, et réalise également le moulage de pièces en plastique. Les produits vendus sont parfois des commandes spécifiques de clients, ou parfois des solutions directement conçues par l'entreprise.

Le mode opératoire est le même, que ce soit pour l'aluminium ou le plastique : on conçoit le moule ainsi que le processus précis de moulage, puis on sort quelques pièces "test". Quand on obtient un résultat satisfaisant, on passe à la production de masse, les produits étant souvent vendus en grandes quantités.

Entre les quelques fonderies dans la province du Cap Occidental la compétition est grande, et la meilleure méthode pour se démarquer est l'innovation. La complexité du processus n'a pas complètement atteint la haute technique utilisée dans les pays les plus développés, donc il y a toujours des aspects à améliorer.

Mon but a été d'**automatiser le processus de moulage**, en améliorant la table de moulage. Mon temps fut environ équitablement réparti dans trois espaces de travail au sein des locaux : **La fonderie** où j'ai commencé par observer, puis plus tard effectué des tests, **les bureaux** où s'est déroulée toute la partie étude / programmation, et enfin **l'atelier de mécanique,** dans lequel j'ai mis en place le système.

Automatisation de la Table de Moulage

Introduction

Les tables de moulage utilisées à Ajax possèdent différentes caractéristiques, pour répondre à différents besoins. Cependant elles ont toutes en commun : Deux massives plaques fixes, entre lesquelles on déplace une plaque mobile à l'aide d'un ou deux vérins hydrauliques. En ce qui concerne la table sur laquelle j'ai travaillé, elle était au début pilotée par deux boutons, pour déplacer la plaque dans un sens ou dans l'autre ; ainsi fermant ou ouvrant le moule qui est installé à l'intérieur.

L'importance de l'automatisation du processus de moulage réside surtout dans la répétabilité précise des actions. En réduisant l'aspect variable des actions humaines, il suffit de trouver les bons paramètres qui donnent une pièce de qualité ; on est ensuite capables de réitérer le processus avec précision. Cela permet aussi de réguler le temps de cycle, donc de fixer le rendement (en nombre de pièces produites par heure) de la machine.

Afin d'automatiser la table, j'ai commencé par **étudier le processus de moulage,** en détail, pour déterminer les solutions possibles. L'étape suivante fut l'**installation du matériel**, puis finalement la **programmation de l'automate** et de l'écran d'information.

1. Étude du Processus de Moulage

La première chose à faire fut simplement d'observer le travail des ouvriers de la fonderie, pour connaître la séquence exacte des mouvements de la table de moulage, pour un cycle complet. Voici ce que j'en ai tiré :

a. Chauffer l'intérieur du moule jusqu'à la **température de départ T_0**. Pour cela on ouvre légèrement le moule (5cm à 10cm) et on installe un chalumeau orienté vers les surfaces intérieures du moule.

b. Vaporiser les surfaces intérieures avec les deux types de **revêtement** : le revêtement blanc favorise le transfert de chaleur de l'aluminium au moule pour qu'il se solidifie plus vite, et le revêtement noir diminue l'adhérence de la pièce à la surface vaporisée. Ensuite on chauffe à nouveau le moule jusqu'à T_0.

c. À ce moment on commence les moulages : on **ferme le moule** pour y **verser l'aluminium**, puis on attend (5s à 15s) la solidification.

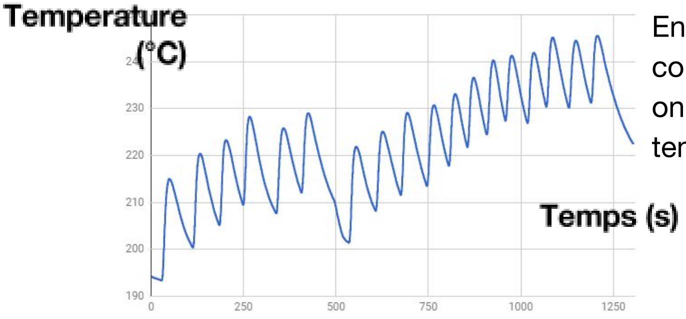
d. On peut **ouvrir le moule** pour **récupérer la pièce**. Dans l'idéal, la pièce se détache de la "moitié fixe" du moule, et part avec la moitié mobile. Cependant il arrive assez souvent que la pièce reste attachée aux deux moitiés. Dans ce cas, on doit ouvrir / fermer / ouvrir / fermer / etc…, et la pièce se décroche comme on le souhaite.

e. On **ouvre le moule au maximum**, et deux tiges métalliques poussent l'arrière du moule. Cela pousse les goupilles d'éjection à travers le moule, et la pièce se retire naturellement du moule.

f. Pour finir on referme le moule complètement, pour rentrer les goupilles d'éjection, et on recommence ensuite pour la pièce suivante.

Les premières pièces ne seront pas bonnes : au début le moule n'est pas assez chaud, et des défauts de moulage apparaissent. Après quelques cycles (5 à 10), la qualité des pièces devient optimale car la température se stabilise.

Ici on voit l'évolution de la température pendant les moulages :



Les résultats du premier test

En regardant l'évolution des pics successifs, correspondant aux versages d'aluminium liquide, on constate bien une phase d'augmentation de la température suivie d'une phase de stabilisation. *(L'interruption survenant après 6 cycles est seulement due à une pause pour vaporiser une couche supplémentaire de revêtement)*

De ces résultats, j'ai décidé de garder **3 valeurs clés**, spécifiques à chaque moule :

- La température à laquelle commencer les moulages, à atteindre avec le chalumeau. C'est la **température de départ T_0**.
- Le moment auquel il faut ouvrir le moule. Cela correspond à une chute d'un certain nombre de °C après un pic. C'est le **delta de température ΔT**.
- Et enfin la température que les pics doivent atteindre, pour pouvoir considérer qu'on est en régime permanent (phase à laquelle les pièces seront de qualité optimale). C'est la **température cible T_1**.

2. Installation du matériel

La table étant déjà en utilisation avant le début de mon stage, les composants nécessaires à son utilisation basique étaient déjà installés : Les deux **plaques fixes**, la **plaque mobile**, deux **vérins hydrauliques** qui fonctionnent comme une vérin unique, et les deux systèmes (un pour fermer, l'autre pour ouvrir) **<Bouton - Relais - Valve>** qui contrôlent le système hydraulique. Tout ça fonctionne sur 24V, fournis par le transformateur 230V ⇒ 24V, et pouvant être coupés par le bouton d'arrêt d'urgence.

Les relais sont de simples dispositifs 1 entrée / 1 sortie :

- Entrée ON ⇒ Sortie ON
- Entrée OFF ⇒ Sortie OFF

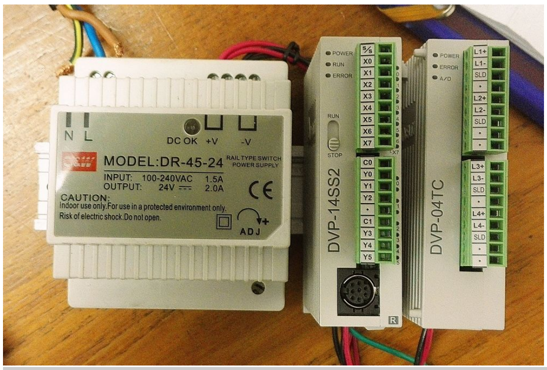
Automatiser ce système nécessitait forcément de remplacer ces relais par un dispositif programmable, en l'occurrence un **PLC** (Programmable Logic Controller) qui possède 8 entrées / 8 sorties.

J'ai également rajouté des équipements nécessaires pour informer le PLC de l'état du système : deux **capteurs de pression** pour détecter les fins de courses des vérins, un **capteur de fin de course**, installé de manière à pouvoir l'ajuster selon les besoin, un **commutateur 3 modes** pour permettre à l'opérateur de choisir entre les modes ***Manuel, Semi-Automatique, Automatique***, et un **commutateur à clé** pour l'usage administrateur.

Pour la température, le fournisseur habituel d'Ajax propose des thermocouples personnalisés. J'ai choisi des **thermocouples de type J** (linéaires jusqu'à **750°C**) et résistants jusqu'à 800°C. Chaque thermocouple est associé à un moule spécifique, car le **comportement thermique** que l'on obtient dépend sensiblement de la manière dont la sonde est mise dans le moule ; et on veut des résultats cohérents d'un jour à l'autre.

Un module spécial pour traiter la température a été assemblé au PLC, c'est à ce module qu'on branche le thermocouple. La résolution est de **0.1°C**.

Il est impensable de rendre la table de moulage autonome sans l'équiper d'une interface pour communiquer avec l'opérateur ; l'intégration d'un **écran** était nécessaire. L'écran choisi dispose d'un affichage **240x128** pixels, résolution suffisante pour afficher des instructions et une courbe de température, et de multiples **boutons** pour permettre à l'opérateur de taper les caractéristiques du moule (T_0, T_1 et ΔT).



Transformateur, PLC et Module de Temperature

L'écran : affichage + pavé numérique

3. Programmation de l'automate

Pour des raisons de sécurité, l'opérateur doit pouvoir prendre le contrôle total de la machine à n'importe quel moment. Grâce au **commutateur 3 modes** câblé au PLC, il peut choisir entre 3 modes : Manuel, Semi-Automatique, Automatique.

Le **mode manuel** donne le même résultat que la configuration qui existait déjà, avec les relais : il faut **laisser appuyer** sur un bouton pour effectuer le mouvement correspondant (ouvrir ou fermer).

En **mode semi-automatique**, il n'a qu'à **appuyer une fois** sur un bouton pour entamer le mouvement correspondant, et sur l'un des deux boutons pour l'arrêter. Sans tenir compte de la température, c'est juste plus pratique pour l'opérateur.

Le **mode automatique** était l'objectif du stage. Le PLC ouvre / ferme le moule en suivant le cycle vu en *1. Étude du Processus de Moulage*. J'ai effectué le programme sur le logiciel fourni avec le PLC, qui fonctionne en langage **Ladder** ; le programme détaillé est en annexe (**Appendix A**).

Le but est de réduire le nombre d'interventions de l'opérateur, même si certaines opérations doivent quand même être effectuées à la main : verser l'aluminium, et retirer la pièce du moule. Les 4 caractéristiques de chaque moule (I.D., T_0, T_1 et ΔT) sont stockées dans la mémoire morte du PLC. Il y a 1920 emplacements disponibles, donc on peut stocker 1920/4 = 480 moules différents.

Le mode automatique prend en compte la température en temps réel, et affiche les instructions du mode opératoire sur l'écran.

L'écran est lui-aussi fourni avec un logiciel de programmation. Il permet de créer des pages de résolution **240x128**, et de rediriger l'opérateur d'une page à l'autre si la condition choisie devient vraie.

À l'allumage, on demande à l'opérateur s'il veut enregistrer un nouveau moule (en entrant un I.D. et les trois valeurs caractéristiques) ou alors en utiliser un pré-existant. Vient ensuite la page correspondant au mode de fonctionnement, qui affiche la courbe de température en temps réel, et les instructions si le mode est automatique.

Pour l'**étude de température** au début du stage, et notamment pour obtenir la courbe de température vue en *1. Étude du Processus de Moulage*, il a fallu écrire un tout autre **programme Ladder** dans le PLC afin d'envoyer les valeurs de températures vers un ordinateur. Avec le câble RS-232 fourni, les informations sont envoyées en ASCII, ce qui limite la capacité des données. Une valeur de température doit donc être codée sur **2 bytes ASCII**, et être reçue dans un fichier excel muni d'un **programme VBA** pour décoder l'information ; les deux programmes Ladder et VBA sont détaillés en annexe (**Appendix B**).

Conclusion

Durant les premiers jours, je commencais à peine à comprendre l'étendue du travail qui m'était attribué. Pendant les derniers jours, j'ai dû préparer la documentation pour la prochaine personne à reprendre ce travail. Dans tous les cas, du début à la fin, j'ai compris que de nombreuses qualités, autres que des compétences techniques, étaient nécessaires pour remplir correctement une mission en entreprise.

En ce qui concerne l'organisation de l'entreprise, j'ai retrouvé des aspects déjà observés dans des stages précédents. Par exemple, le besoin d'enregistrer et de suivre chaque achat professionnel, d'attribuer un numéro à chaque nouvelle tâche, etc…

En revanche cette fois-ci, le fait d'être acteur de ce système rend les enjeux complètement différents et m'a forcé à me plier à ce mode de fonctionner. Cela a aussi évolué au cours des trois mois de stage : des indications de sécurité et des instructions ont été installées près des lieux de travail dangereux, comme les fournaises à aluminium.

Une stagiaire en ressources humaines a également rendu plus facile pour les ouvriers de se faire entendre. Parfois les relations hiérarchisées donnent lieux à des tensions au sein d'une équipe, et un service dédié à ce genre de problèmes s'avère indispensable.

L'aspect technique du stage fut fortement enrichissant, et c'est dû à l'apprentissage constant qu'il m'a fallu suivre. J'avais à ma disposition un bon nombre de machines pour scier, meuler, souder, poncer, etc… Je suis devenu autonome pour concevoir, fabriquer une pièce et l'intégrer à la table de moulage.

Il s'est avéré essentiel de toujours garder une **vue d'ensemble** sur ce que je faisais. Il est utile de diviser un problème en plusieurs problèmes plus simples, mais pas si on oublie de penser au système dans son ensemble. C'est du moins ce que je retiens d'un projet qui mélange des aspects aussi bien mécaniques que electroniques.

Le thème de ce stage s'inscrit vraiment dans la continuité de ce qui m'a intéressé durant mes années d'école : travailler à des solutions pour créer ou améliorer un système mécatronique. De plus, pendant ce stage j'ai été accepté dans le master recherche de mon choix, qui est également tourné vers la robotique. Sachant que ce que je faisais me préparait à la suite de mes études, j'ai été d'autant plus motivé à travailler sur ce projet.