Algerian Journal of Research and Technology
https://doi.org/10.58681/ajrt.22060101
*A.J.R.T Volume 6 (N°1) (2022) 1 – 10*
Selection and peer-review under responsibility of the scientific committee of The 1st International Symposium on INDUSTRIAL ENGINEERING, MAINTENANCE AND SAFETY IEMS'22.

# Design and Realization of an Automated Drug Distribution System Based on Arduino

*S. Bouzidi [1], A. Tabti [1], K. Hachemi [1]*

*[1] Institute of Maintenance and Industrial Safety, University of Oran 2, Algeria.*

**Corresponding author.* *sarrabouzidi95@gmail.com, sammatbt195@gmail.com, hachemi.khalid1@gmail.com*

Published. April 15, 2022.

**DOI:** https://doi.org/10.58681/ajrt.22060101

**Abstract.** This paper presents the design and the realization of an automated drug dispensing prototype system, of 16 types, this system was inspired by the French automaton 'ApotéKa' which uses the Free Fall flow-rack AS/RS (FF-flow-rack AS / RS). The system realized is based on servomotors to provoke the free fall of the drugs towards a conveyor that transports them. A PCA968 module, which is controlled by Arduino Mega2560 through PC, the presence, pilots the servomotors, or an infrared obstacle sensor detects the absence of the boxes in the bins.

***Keywords.*** Flow-rack AS/RS, Free Fall, Drug distribution, Servomotors, Arduino, control.

## INTRODUCTION

Automation is developing more and more within pharmacies. Automated drug dispensing systems (ADDSs) have been utilized in various pharmacies over the last ten years (García et al., 2017). For instance, Over the past decade, the use of this technology in hospital pharmacies has been on the rise with 97% of US hospital pharmacies using ADS as of 2014 (Pedersen et al., 2014). Due to the various advantages offered by these systems, in the paper of Franklin et al. (2008), the authors reported that ADDSs are substituting a few regular, time-consuming filling practices by increasing precision, enhancing productivity, and reducing medication errors. Indeed, despite a crucial period for the profession, during which the reduction of costs is the main concern of the incumbents. Automation companies do their best proposing solutions adapted to all types of pharmacies; small, medium, large, or shopping centers (with their large stock width). At certain points, they knew how to adapt to the growing needs of pharmacists moving toward more autonomous systems (automatic boxes, storage), more compact and less expensive, for that several technological solutions have appeared to automate drug dispensing,

such as automated storage and retrieval system (AS/RS), with new variant free- fall-flow-rack AS/RS (APOTEKA) which introduced by Mekapharm (2016).
The work method (storage and retrieval operation) of the FF-flow-rack AS/RS is as follows: The storage operation, which is performed by a storage machine or an operator, where the products are handled from the pick-up station to store in the adequate bin. The retrieval operation consists of two phases: The first one is the ejection of the product, which is achieved by the excitement of the electromagnet of the bin containing the desired product. This will cause the free fall of the product on the transport conveyor. The second phase is the transport of this product by the conveyor until reaching the drop-off station (Metahri and Hachemi, 2018).
For making the flow-rack AS/RS more effective much research has been done, (Sari et al., 2005) presented mathematical models for the expected travel time for a flow-rack AS/RS, which used two (S/R) machines, After that, Sari (2010) performed a comparative study between unit-load AS/RS and flow-rack AS/RS. The author considered two comparison parameters: space‘s use and travel time, and to decrease the travel time for the flow-rack AS/RS many researchers have proposed various methods, in the paper of Hachemi and Alla (2008) authors presented an optimization method of retrieval sequencing where the AS/RS was depicted by a colored Petri net model. By considering a dual cycle coupled with a lower mid-point S/R crane dwell-point policy, Xu et al. (2018) have developed a continuous travel-time model for DC in 3D compact storage system; then, he has used the analytical models to optimize system dimensions.
The free-fall flow rack AS/RS offers many advantages for example the reduction of average retrieval–travel time can reach 38% when using the FF flow rack (Metahri and Hachemi, 2018), this automatic system was put in place to allow pharmacies to win time preparing prescriptions and pick up the orders thereafter. It also reduces errors knowing that dispensing errors are common, often go undetected, and can have serious consequences (Simpson, 2008), Two studies (Schwarz and Brodowy, 1995; Kratz and Thygesen, 1992) on pharmacy-based automatic dispensing devices showed a decrease in dispensing errors.
The presented system is a practical study that explains the design and realization of an automated drug distribution system which is an AS / RS free-fall flow-rack, with optimal costs when based on the Arduino. The rest of the paper is organized as follows: In Section 2, we present our system description, programming in Section 3, the Final realization of the system in Section 4, and the conclusion in section 5.

## SYSTEM DESCRIPTION

The French automaton, ‘ApotéKa’, inspired our system; we chose to pilot it with Arduino which is less expensive than a programmable logic controller. We made a prototype system to ensure the distribution of 16 types of medication (16 lockers), automatically by a computer, which consists of sending the data via the serial port to the Arduino in the form of a data frame containing the necessary information such as the name of the locker and the number of boxes to distribute. From this data frame, the Arduino controls servomotors (provoke drug to descend in free fall) through an interface, the conveyor is also run to finalize the operation of the Distribution.

### System components

To achieve our goal, we used different equipment and materials:
- A flow rack about 77 cm high and 50 cm deep, of 16 types of medication (16 lockers), grouping together a set of products. The system contains four trays per line with different lengths depending on the type of medication: (Locker 1: 10 cm length, Locker 2: 12 cm length, Locker 3: 13 cm length, Locker 4: 12 cm length). Regarding the angle of inclination of the

lockers, we chose 30 degrees, allowing rapid drug sliding to make the system more efficient (Fig. 1).

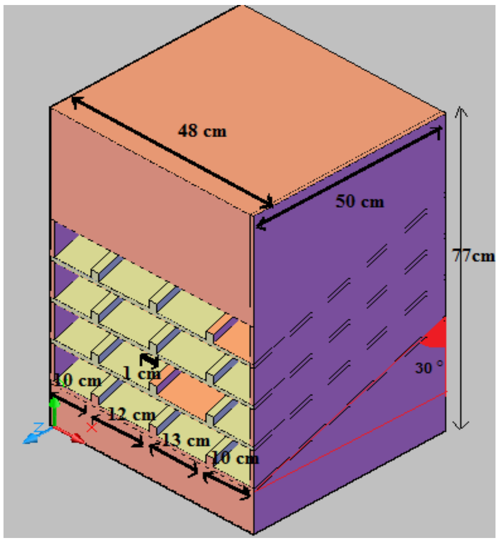
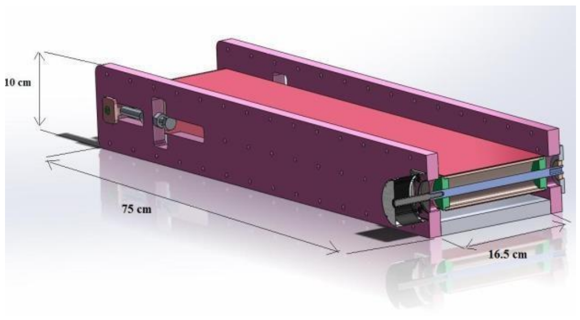


Fig.1. Flow rack design in 3D.

- Conveyor, which is necessary to bring medication near counters, is a conveyor belt type (Fig. 2, Fig. 3).



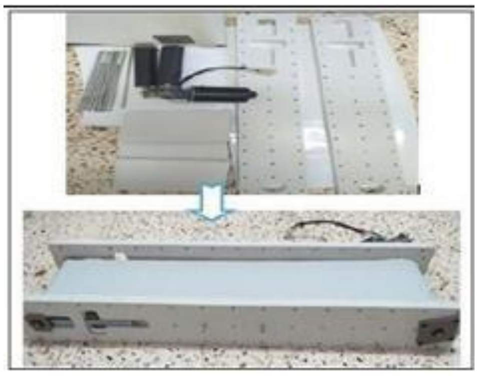


Fig.2. Conveyor design in 3D. Fig.3. Conveyor in real picture.

Running and stopping of the conveyor are controlled by a DC motor, and must relate to 1 channel 5v relay module controlling the motor which has a voltage of 12 v much higher than what the Arduino could accept (5v).
- Servomotors (Sg90) which rotate at a determined angle cause the free fall of medication, we chose this component for two reasons: the precision of rotation, it will be able to actuate heavy "limbs" because it is powerful.
- 4 channel infrared obstacle sensor, to find out if a locker requires refilling on a placed infrared obstacle sensor that detects the presence of the box (the channel to be filled is displayed on the control computer).
- Electronic circuit (PCA968 Servo Driver + Arduino mega), The PCA9685 module is a better solution which is to using a separate servo driver board. This will offload the task of sending PWM messages to the servos, which will allow our Arduino to do better things on the one hand; on the other hand, we avoid the electrical risks that can destroy the Arduino board. The PCA9685 is powered by an external voltage source, for this, we used an ‘AC 100-240v 50/60 Hz’ power supply of 5v / 2 A.
- Cables type of ‘22 AWG - 4 p’ to establish connections between the elements of the system.

## Connections and simulation

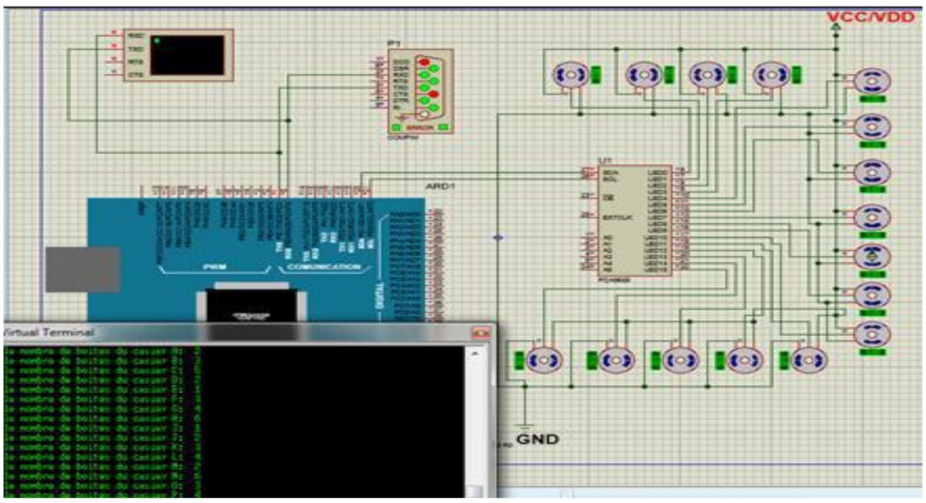
### a. Arduino board with PCA9685 and servomotors

We used the PCA9685 board to control the 16 servomotors since the PCA9685 card contains the I2C bus, which shares the same SCL / SDA signals. The wiring is relatively simple, we have connected the 16 servomotors on the 16 ports of the PCA9685 controller and the latter has connected to Arduino on:

- Digital20: with SDA
- Digital 21: with SCL to establish an I2C communication.
- +5V: with Vcc
- GND: with GND to have a common ground between the PCA9685 module and Arduino.

The logic of the PCA9685 is supplied via the VCC pin (+ 5V). If the latter is not connected, the logic part of the circuit is not powered, so there is no control circuit.

We simulated our wiring diagram in PROTEUS which allowed us to observe if there are any faults in the connection and to be able to remedy any problems that we can face without danger, as schematically indicated in figure 4.

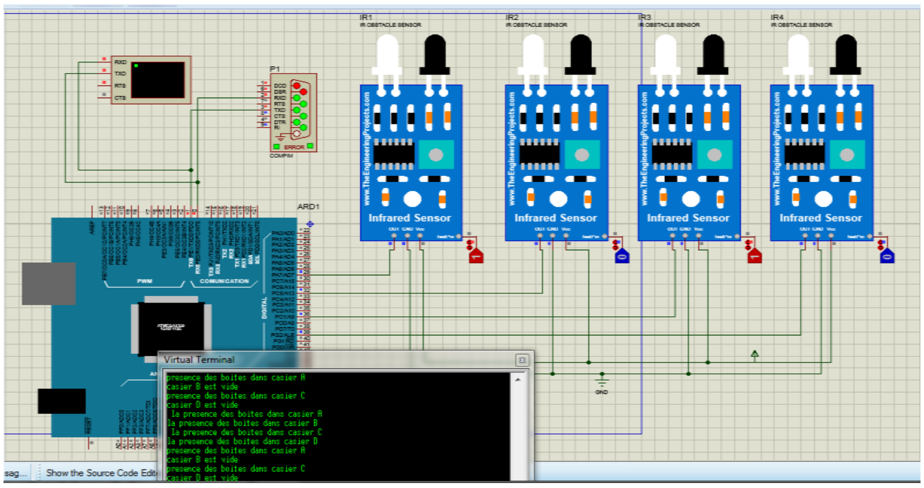


```
le nombre de boites du casier A: [illegible]
le nombre de boites du casier B: [illegible]
le nombre de boites du casier C: [illegible]
le nombre de boites du casier D: [illegible]
le nombre de boites du casier E: [illegible]
le nombre de boites du casier F: [illegible]
le nombre de boites du casier G: [illegible]
le nombre de boites du casier H: [illegible]
le nombre de boites du casier I: [illegible]
le nombre de boites du casier J: [illegible]
le nombre de boites du casier K: [illegible]
le nombre de boites du casier L: [illegible]
le nombre de boites du casier M: [illegible]
le nombre de boites du casier N: [illegible]
le nombre de boites du casier O: [illegible]
le nombre de boites du casier P: [illegible]
```

Fig.4. The Simulation in PROTEUS.

### b. Arduino board with a 4-channel infrared obstacle sensor

The sensor connection with Arduino is like the following in figure 5.



```
presence des boites dans casier A
casier B est vide
presence des boites dans casier C
casier D est vide
 la presence des boites dans casier A
la presence des boites dans casier B
 la presence des boites dans casier C
la presence des boites dans casier D
presence des boites dans casier A
casier B est vide
presence des boites dans casier C
casier D est vide
```

Fig.5. The connection between Arduino with the sensor in PROTEUS.

Table 1 shows the connection of the transmitter module with the receiver:

Table 1. Connection transmitter / receiver

| The Transmitter | The receiver |
|---|---|
| Vcc | Vcc |
| GND | GND |
| OUT | OUT |

Table 2 shows the connection of the receiver module with the Arduino:

Table 2. Connection receiver / Arduino

| The receiver | The Arduino |
|---|---|
| Vcc | +5v |
| GND | GND |
| OUT1 | Digital OUT |
| OUT2 | Digital OUT |
| OUT3 | Digital OUT |
| OUT4 | Digital OUT |

### c. The Arduino board with the DC motor and relay

The connection between the Arduino, DC motor, and relay is in figure 6.

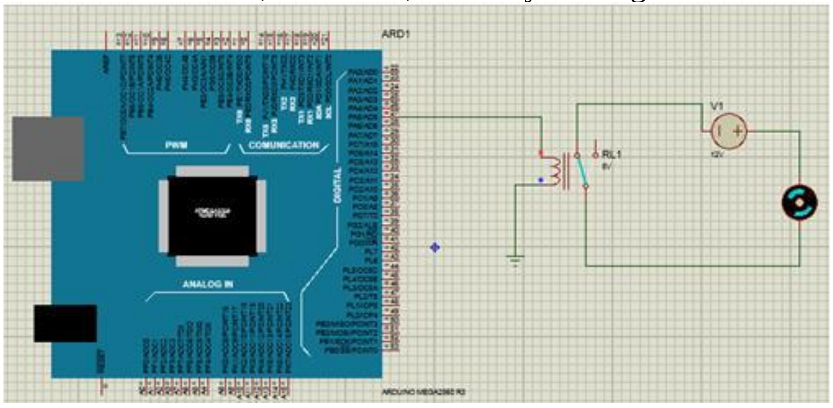


Fig.6. The connection between Arduino, the DC motor, and the relay in PROTEUS.

Table 3 shows the connection of relays with Arduino:

Table 3. Connection relay/Arduino

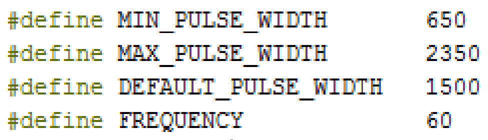
| **Relay** | **Arduino** |
|---|---|
| Vcc | +5V |
| GND | GND |
| OUT | Digital OUT |

**PROGRAMMATION**

To clip the PCA9685 module code with the Arduino we have to use the library **Adafruit_PWMServoDriver.h :** `#include <Adafruit_PWMServoDriver.h>`

To use the library **Adafruit_PWMServoDriver.h,** we must first declare :

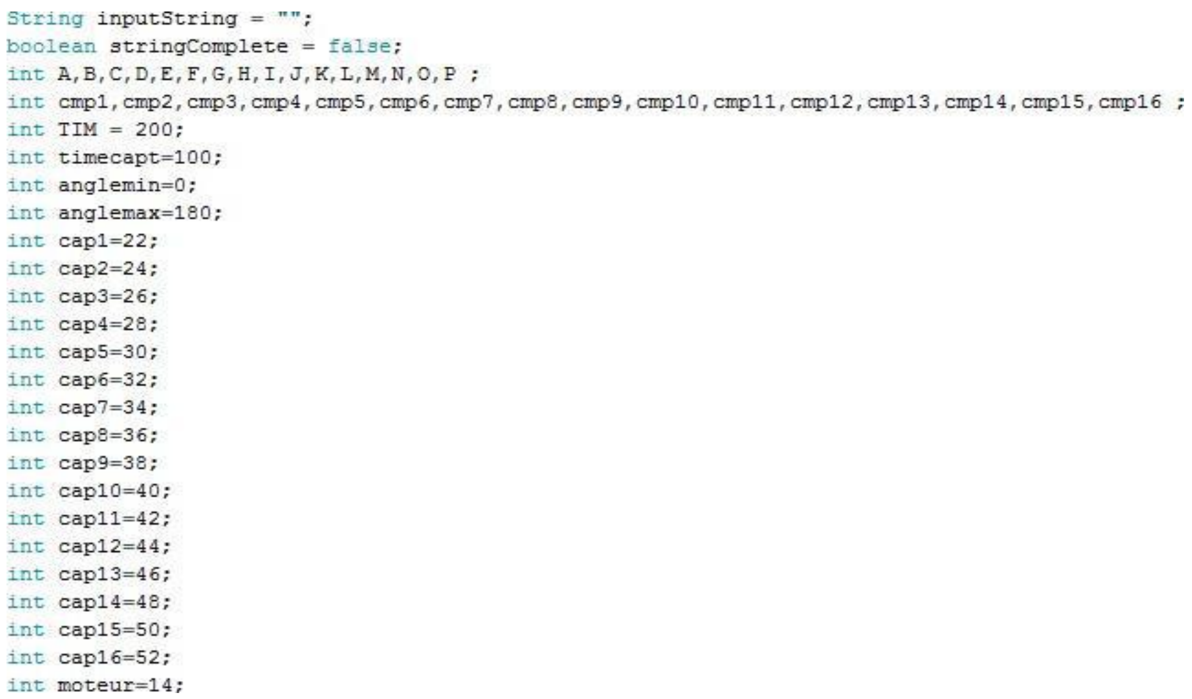
- Minimum and maximum pulse width.
- Default pulse width and frequency (Fig. 7).



```
#define MIN_PULSE_WIDTH       650
#define MAX_PULSE_WIDTH       2350
#define DEFAULT_PULSE_WIDTH   1500
#define FREQUENCY             60
```

Fig.7. Pulse width and frequency declaration.

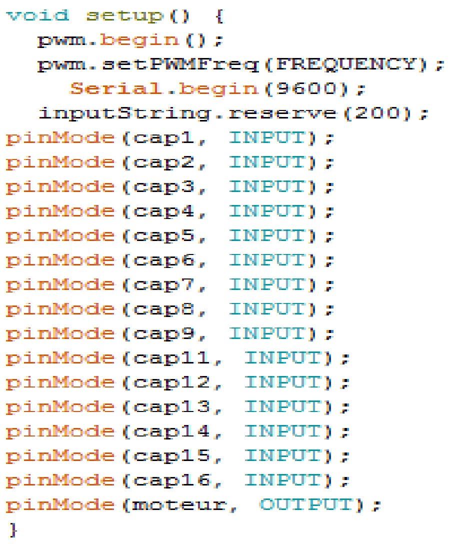
Also the declaration of variable: (Fig. 8)



```
String inputString = "";
boolean stringComplete = false;
int A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P ;
int cmp1,cmp2,cmp3,cmp4,cmp5,cmp6,cmp7,cmp8,cmp9,cmp10,cmp11,cmp12,cmp13,cmp14,cmp15,cmp16 ;
int TIM = 200;
int timecapt=100;
int anglemin=0;
int anglemax=180;
int cap1=22;
int cap2=24;
int cap3=26;
int cap4=28;
int cap5=30;
int cap6=32;
int cap7=34;
int cap8=36;
int cap9=38;
int cap10=40;
int cap11=42;
int cap12=44;
int cap13=46;
int cap14=48;
int cap15=50;
int cap16=52;
int moteur=14;
```

Fig.8. Variable declaration.

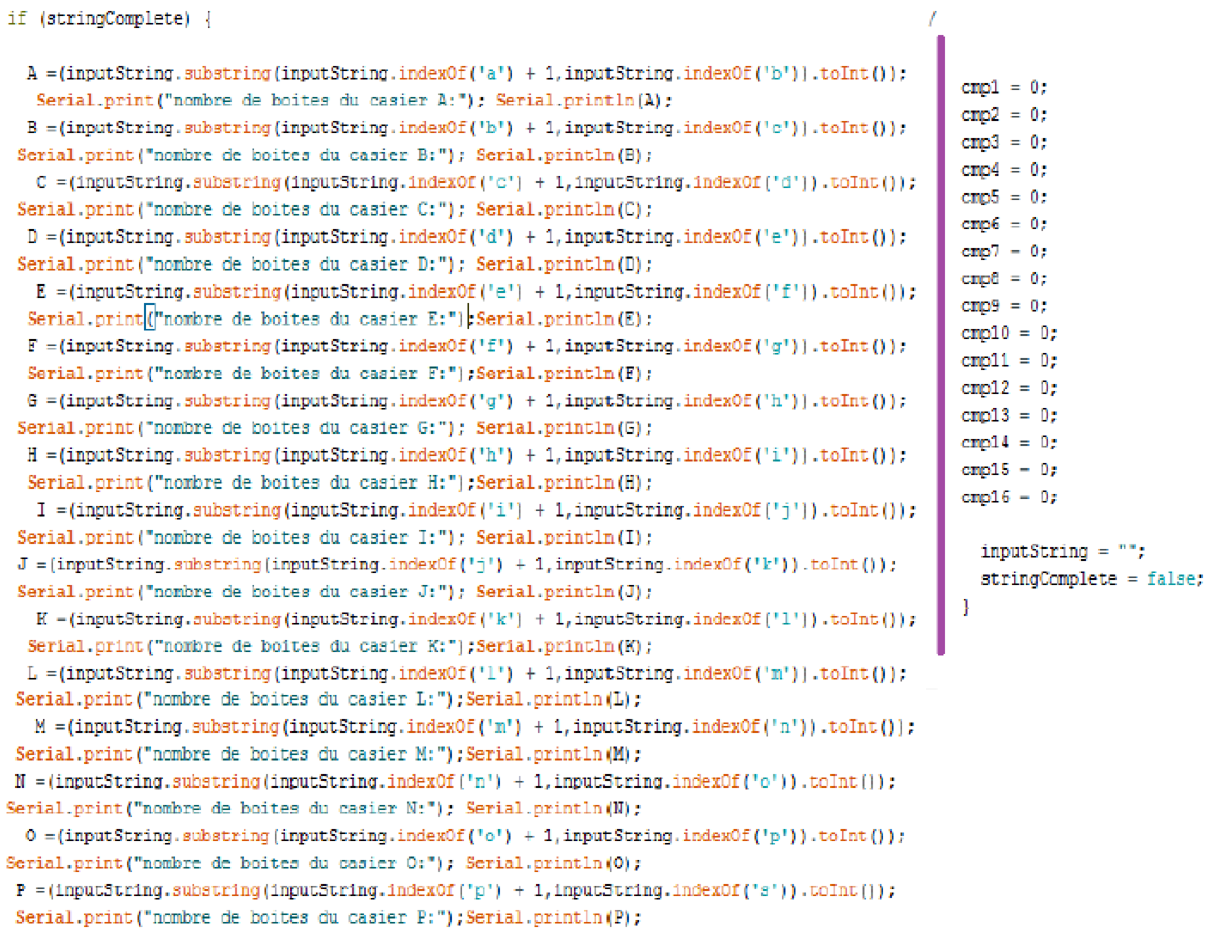
We used the Serial Event function to communicate with the serial monitor, which works by default at 9600 baud, so we have reserved 200 bytes for the string. In addition, we declared the sensors as inputs and the motor as an output. (Fig. 9)



```
void setup() {
  pwm.begin();
  pwm.setPWMFreq(FREQUENCY);
    Serial.begin(9600);
  inputString.reserve(200);
pinMode(cap1, INPUT);
pinMode(cap2, INPUT);
pinMode(cap3, INPUT);
pinMode(cap4, INPUT);
pinMode(cap5, INPUT);
pinMode(cap6, INPUT);
pinMode(cap7, INPUT);
pinMode(cap8, INPUT);
pinMode(cap9, INPUT);
pinMode(cap11, INPUT);
pinMode(cap12, INPUT);
pinMode(cap13, INPUT);
pinMode(cap14, INPUT);
pinMode(cap15, INPUT);
pinMode(cap16, INPUT);
pinMode(moteur, OUTPUT);
}
```

Fig.9. Defining input and output.

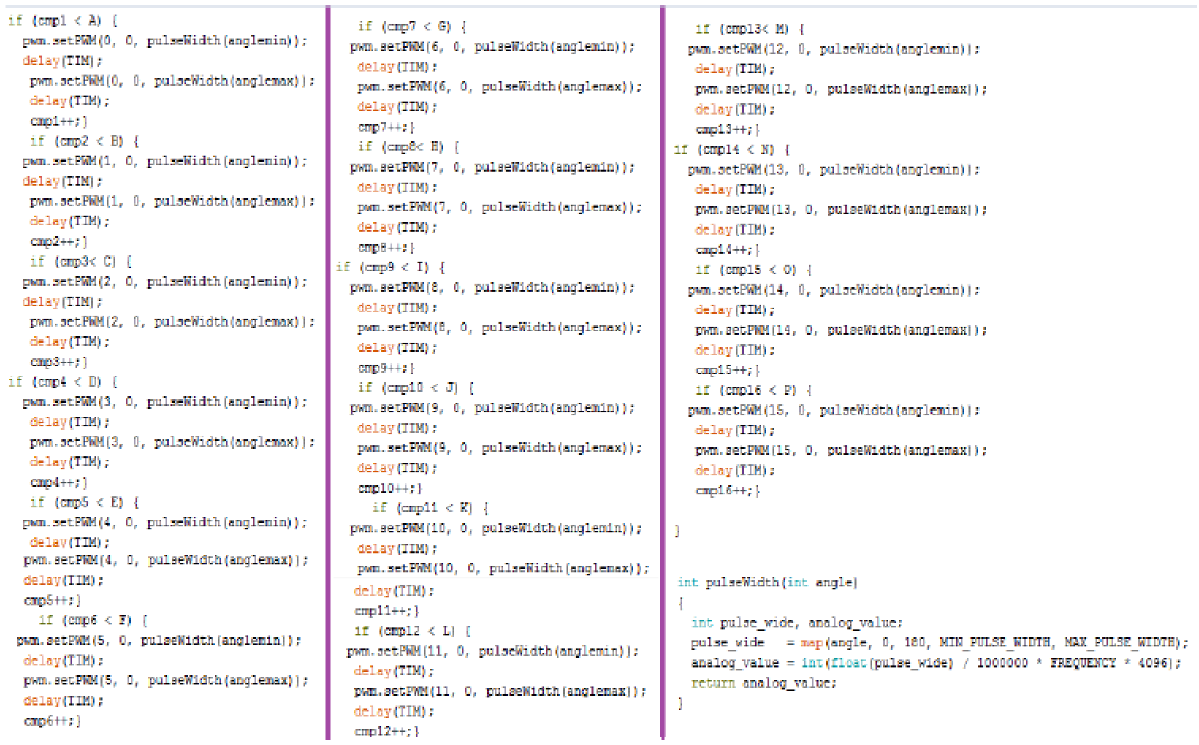
The program has established a connection with the serial monitor to read the received data; it receives a character frame 'a' until 'q' which indicates the name of the locker (servo), and between each alphabet. We find a number is a counter associated with each servo, which indicated the number of servo movements (number of products to distribute) (Fig. 10).



```
if (stringComplete) {

  A =(inputString.substring(inputString.indexOf('a') + 1,inputString.indexOf('b')).toInt());
   Serial.print("nombre de boites du casier A:"); Serial.println(A);
  B =(inputString.substring(inputString.indexOf('b') + 1,inputString.indexOf('c')).toInt());
 Serial.print("nombre de boites du casier B:"); Serial.println(B);
   C =(inputString.substring(inputString.indexOf('c') + 1,inputString.indexOf('d')).toInt());
 Serial.print("nombre de boites du casier C:"); Serial.println(C);
  D =(inputString.substring(inputString.indexOf('d') + 1,inputString.indexOf('e')).toInt());
 Serial.print("nombre de boites du casier D:"); Serial.println(D);
   E =(inputString.substring(inputString.indexOf('e') + 1,inputString.indexOf('f')).toInt());
  Serial.print("nombre de boites du casier E:");Serial.println(E);
  F =(inputString.substring(inputString.indexOf('f') + 1,inputString.indexOf('g')).toInt());
  Serial.print("nombre de boites du casier F:");Serial.println(F);
  G =(inputString.substring(inputString.indexOf('g') + 1,inputString.indexOf('h')).toInt());
 Serial.print("nombre de boites du casier G:"); Serial.println(G);
  H =(inputString.substring(inputString.indexOf('h') + 1,inputString.indexOf('i')).toInt());
  Serial.print("nombre de boites du casier H:");Serial.println(H);
   I =(inputString.substring(inputString.indexOf('i') + 1,inputString.indexOf('j')).toInt());
 Serial.print("nombre de boites du casier I:"); Serial.println(I);
 J =(inputString.substring(inputString.indexOf('j') + 1,inputString.indexOf('k')).toInt());
 Serial.print("nombre de boites du casier J:"); Serial.println(J);
   K =(inputString.substring(inputString.indexOf('k') + 1,inputString.indexOf('l')).toInt());
  Serial.print("nombre de boites du casier K:");Serial.println(K);
  L =(inputString.substring(inputString.indexOf('l') + 1,inputString.indexOf('m')).toInt());
 Serial.print("nombre de boites du casier L:");Serial.println(L);
   M =(inputString.substring(inputString.indexOf('m') + 1,inputString.indexOf('n')).toInt());
 Serial.print("nombre de boites du casier M:");Serial.println(M);
 N =(inputString.substring(inputString.indexOf('n') + 1,inputString.indexOf('o')).toInt());
Serial.print("nombre de boites du casier N:"); Serial.println(N);
  O =(inputString.substring(inputString.indexOf('o') + 1,inputString.indexOf('p')).toInt());
Serial.print("nombre de boites du casier O:"); Serial.println(O);
 P =(inputString.substring(inputString.indexOf('p') + 1,inputString.indexOf('s')).toInt());
 Serial.print("nombre de boites du casier P:");Serial.println(P);

cmp1 = 0;
cmp2 = 0;
cmp3 = 0;
cmp4 = 0;
cmp5 = 0;
cmp6 = 0;
cmp7 = 0;
cmp8 = 0;
cmp9 = 0;
cmp10 = 0;
cmp11 = 0;
cmp12 = 0;
cmp13 = 0;
cmp14 = 0;
cmp15 = 0;
cmp16 = 0;

  inputString = "";
  stringComplete = false;
}
```

Fig.10. Creating the Data frame.

Once the program receives the data frame, each servo will move depending on its counter (Fig. 11).



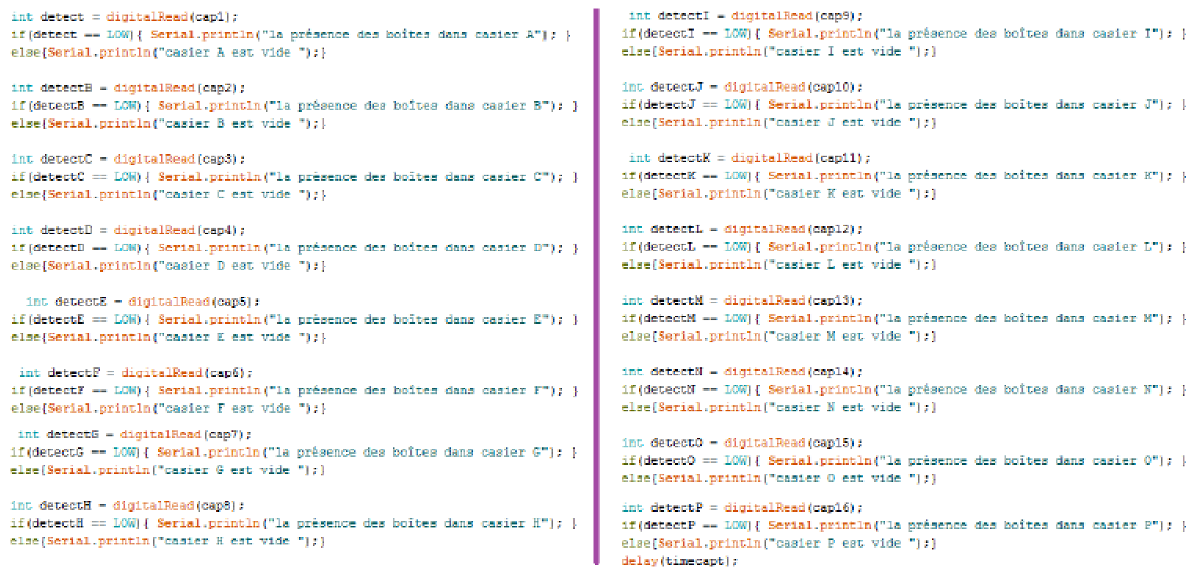
```
if (cmp1 < A) {
  pwm.setPWM(0, 0, pulseWidth(anglemin));
  delay(TIM);
   pwm.setPWM(0, 0, pulseWidth(anglemax));
   delay(TIM);
   cmp1++;}
   if (cmp2 < B) {
  pwm.setPWM(1, 0, pulseWidth(anglemin));
  delay(TIM);
   pwm.setPWM(1, 0, pulseWidth(anglemax));
   delay(TIM);
   cmp2++;}
   if (cmp3< C) {
  pwm.setPWM(2, 0, pulseWidth(anglemin));
  delay(TIM);
   pwm.setPWM(2, 0, pulseWidth(anglemax));
   delay(TIM);
   cmp3++;}
if (cmp4 < D) {
  pwm.setPWM(3, 0, pulseWidth(anglemin));
   delay(TIM);
   pwm.setPWM(3, 0, pulseWidth(anglemax));
   delay(TIM);
   cmp4++;}
   if (cmp5 < E) {
  pwm.setPWM(4, 0, pulseWidth(anglemin));
   delay(TIM);
  pwm.setPWM(4, 0, pulseWidth(anglemax));
  delay(TIM);
  cmp5++;}
    if (cmp6 < F) {
 pwm.setPWM(5, 0, pulseWidth(anglemin));
  delay(TIM);
  pwm.setPWM(5, 0, pulseWidth(anglemax));
  delay(TIM);
  cmp6++;}
```

```
   if (cmp7 < G) {
  pwm.setPWM(6, 0, pulseWidth(anglemin));
   delay(TIM);
   pwm.setPWM(6, 0, pulseWidth(anglemax));
   delay(TIM);
   cmp7++;}
   if (cmp8< H) {
  pwm.setPWM(7, 0, pulseWidth(anglemin));
   delay(TIM);
   pwm.setPWM(7, 0, pulseWidth(anglemax));
   delay(TIM);
   cmp8++;}
if (cmp9 < I) {
  pwm.setPWM(8, 0, pulseWidth(anglemin));
   delay(TIM);
   pwm.setPWM(8, 0, pulseWidth(anglemax));
   delay(TIM);
   cmp9++;}
   if (cmp10 < J) {
  pwm.setPWM(9, 0, pulseWidth(anglemin));
   delay(TIM);
   pwm.setPWM(9, 0, pulseWidth(anglemax));
   delay(TIM);
   cmp10++;}
     if (cmp11 < K) {
  pwm.setPWM(10, 0, pulseWidth(anglemin));
   delay(TIM);
   pwm.setPWM(10, 0, pulseWidth(anglemax));
  delay(TIM);
  cmp11++;}
  if (cmp12 < L) {
 pwm.setPWM(11, 0, pulseWidth(anglemin));
  delay(TIM);
  pwm.setPWM(11, 0, pulseWidth(anglemax));
  delay(TIM);
  cmp12++;}
```

```
   if (cmp13< M) {
  pwm.setPWM(12, 0, pulseWidth(anglemin));
   delay(TIM);
   pwm.setPWM(12, 0, pulseWidth(anglemax));
   delay(TIM);
   cmp13++;}
if (cmp14 < N) {
  pwm.setPWM(13, 0, pulseWidth(anglemin));
   delay(TIM);
   pwm.setPWM(13, 0, pulseWidth(anglemax));
   delay(TIM);
   cmp14++;}
   if (cmp15 < O) {
  pwm.setPWM(14, 0, pulseWidth(anglemin));
   delay(TIM);
   pwm.setPWM(14, 0, pulseWidth(anglemax));
   delay(TIM);
   cmp15++;}
   if (cmp16 < P) {
  pwm.setPWM(15, 0, pulseWidth(anglemin));
   delay(TIM);
   pwm.setPWM(15, 0, pulseWidth(anglemax));
   delay(TIM);
   cmp16++;}

}

int pulseWidth(int angle)
{
  int pulse_wide, analog_value;
  pulse_wide   = map(angle, 0, 180, MIN_PULSE_WIDTH, MAX_PULSE_WIDTH);
  analog_value = int(float(pulse_wide) / 1000000 * FREQUENCY * 4096);
  return analog_value;
}
```

Fig.11. Servomotors move depending on to counter.

To order the sensors we used the instruction digital Read (cap1), this instruction reads either a high level (« 1 » or HIGH) in the case of the presence of boxes in the locker, either a low level (« 0 » or LOW) the case of the locker is empty (Fig. 12).



```
int detect = digitalRead(cap1);
if(detect == LOW){ Serial.println("la présence des boites dans casier A"); }
else{Serial.println("casier A est vide ");}

int detectB = digitalRead(cap2);
if(detectB == LOW){ Serial.println("la présence des boîtes dans casier B"); }
else{Serial.println("casier B est vide ");}

int detectC = digitalRead(cap3);
if(detectC == LOW){ Serial.println("la présence des boîtes dans casier C"); }
else{Serial.println("casier C est vide ");}

int detectD = digitalRead(cap4);
if(detectD == LOW){ Serial.println("la présence des boîtes dans casier D"); }
else{Serial.println("casier D est vide ");}

  int detectE = digitalRead(cap5);
if(detectE == LOW){ Serial.println("la présence des boîtes dans casier E"); }
else{Serial.println("casier E est vide ");}

 int detectF = digitalRead(cap6);
if(detectF == LOW){ Serial.println("la présence des boîtes dans casier F"); }
else{Serial.println("casier F est vide ");}
 int detectG = digitalRead(cap7);
if(detectG == LOW){ Serial.println("la présence des boîtes dans casier G"); }
else{Serial.println("casier G est vide ");}

int detectH = digitalRead(cap8);
if(detectH == LOW){ Serial.println("la présence des boîtes dans casier H"); }
else{Serial.println("casier H est vide ");}
```

```
 int detectI = digitalRead(cap9);
if(detectI == LOW){ Serial.println("la présence des boîtes dans casier I"); }
else{Serial.println("casier I est vide ");}

int detectJ = digitalRead(cap10);
if(detectJ == LOW){ Serial.println("la présence des boites dans casier J"); }
else{Serial.println("casier J est vide ");}

 int detectK = digitalRead(cap11);
if(detectK == LOW){ Serial.println("la présence des boites dans casier K"); }
else{Serial.println("casier K est vide ");}

int detectL = digitalRead(cap12);
if(detectL == LOW){ Serial.println("la présence des boîtes dans casier L"); }
else{Serial.println("casier L est vide ");}

int detectM = digitalRead(cap13);
if(detectM == LOW){ Serial.println("la présence des boites dans casier M"); }
else{Serial.println("casier M est vide ");}

int detectN = digitalRead(cap14);
if(detectN == LOW){ Serial.println("la présence des boîtes dans casier N"); }
else{Serial.println("casier N est vide ");}

int detectO = digitalRead(cap15);
if(detectO == LOW){ Serial.println("la présence des boites dans casier O"); }
else{Serial.println("casier O est vide ");}

int detectP = digitalRead(cap16);
if(detectP == LOW){ Serial.println("la présence des boîtes dans casier P"); }
else{Serial.println("casier P est vide ");}
delay(timecapt);
```

Fig.12. Sensor ordering.

To control the conveyor (the DC motor) we must use the instruction digital Write that sends either 5V (HIGH) on the spindle or 0V (LOW).
To operate the system, several adjustments, installations, and configurations are necessary, among these we mention:
- Installation of servomotors.

- Installation of sensors: The sensors must be installed in the designated places (next to the servo motor) to detect the presence of the box in the lockers, then they must be connected to the Arduino.- Installation of the conveyor.

After installing the electronic boards, sensors, modules, and actuators, it is necessary to make the connection with the Arduino board.

After completing the assembly of the model, the testing phase is necessary to ensure the proper functioning of the system. We sent the data via the serial port to the Arduino as a frame containing the necessary information (the name of the locker and the number of boxes dispensed). The servomotors are actuated and the number of boxes in each bin is shown in the serial monitor window as well as the detection results by the sensors, the real picture of the overall system are shown in figure 13.

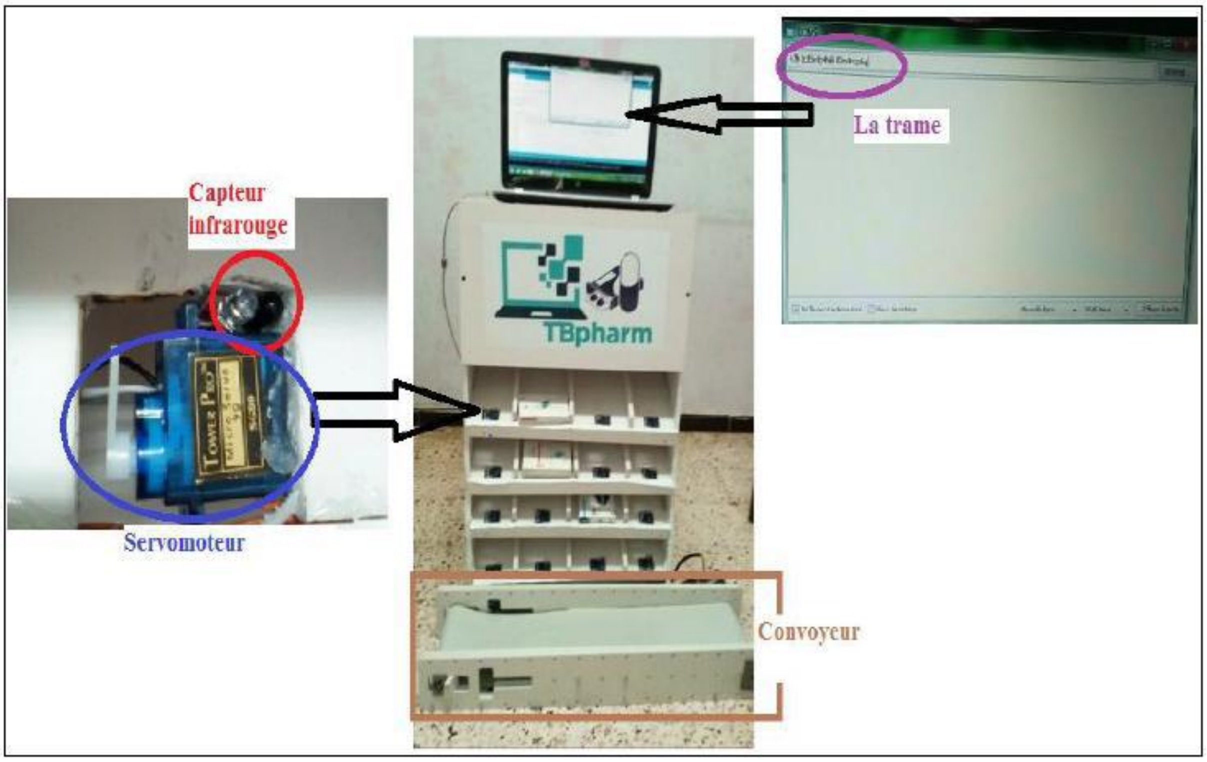


Fig.13. A real picture of the system is realized.

## CONCLUSION

This paper has presented a practical study consists the conception and design of an automated drug dispensing prototype system, We could develop a program that controls by computer 16 servomotors driven using a PCA9685 module to cause the drugs to free fall to the conveyor. Then, we have installed infrared obstacle sensors to detect whether or not boxes are in the lockers, controlled and piloted a conveyor.

We looked to minimize the costs of realization of an automated drug dispensing system using an Arduino control instead of programmable logic controller that costs more. The interest of this practical study was double: On the one hand, it will constitute a realistic prototype allowing demonstrations for professionals who would like to acquire this system, and on the other hand, it will serve as a didactic model for students in practical work in automation and human-machine interface design.

## REFERENCES

Franklin, B. D., O'Grady, K., Voncina, L., Popoola, J., & Jacklin, A. (2010). An evaluation of two automated dispensing machines in UK hospital pharmacy. *International Journal of Pharmacy Practice*, *16*(1). https://doi.org/10.1211/ijpp.16.1.0009

García, C. M., Wanders, B. J., Alei, P. E., & Diaz, C. O. (2017). Automated medication adherence system. *US Patent 9,836,583*.

Hachemi, K., & Alla, H. (2008). Pilotage dynamique d'un système automatisé de stockage/déstockage à convoyeur gravitationnel. *Journal Européen Des Systèmes Automatisés*, *42*(5). https://doi.org/10.3166/jesa.42.487-508

Kratz, K., & Thygesen, C. (1992). A comparison of the accuracy of unit dose cart fill with the Baxter ATC- 212 computerized system and manual filling. *Hospital Pharmacy*, *27*(1).

Metahri, D., & Hachemi, K. (2018). Retrieval–travel-time model for free-fall-flow-rack automated storage and retrieval system. *Journal of Industrial Engineering International*, *14*(4). https://doi.org/10.1007/s40092-018-0263-9

Pedersen, C. A., Schneider, P. J., & Scheckelhoff, D. J. (2017). ASHP national survey of pharmacy practice in hospital settings: Prescribing and transcribing-2016. *American Journal of Health-System Pharmacy*, *74*(17). https://doi.org/10.2146/ajhp170228

Sari, Z., Saygin, C., & Ghouali, N. (2005). Travel-time models for flow-rack automated storage and retrieval systems. *International Journal of Advanced Manufacturing Technology*, *25*(9–10). https://doi.org/10.1007/s00170-003-1932-3

Sari, Z. (2010). Performance evaluation of flow-rack and unit load automated storage & retrieval systems. *Proceedings of ISTEC 2010*, 605-615.

Schwarz, H. O., & Brodowy, B. A. (1995). Implementation and evaluation of an automated dispensing system. *American Journal of Health-System Pharmacy*, *52*(8). https://doi.org/10.1093/ajhp/52.8.823

Simpson, K. R. (2008). Medication safety with heparin. In *MCN The American Journal of Maternal/Child Nursing* (Vol. 33, Issue 2). https://doi.org/10.1097/01.NMC.0000313426.04861.12

Xu, X., Gong, Y. (Yale), Fan, X., Shen, G., & Zou, B. (2018). Travel-time model of dual-command cycles in a 3D compact AS/RS with lower mid-point I/O dwell point policy. *International Journal of Production Research*, *56*(4). https://doi.org/10.1080/00207543.2017.1361049