

USING ARDUINO PLC AND ROBOT WITHIN MECHATRONIC SYSTEMS

Liviu Mihai Sima

PhD student, University POLITEHNICA of Bucharest, Faculty of Industrial and Robotics Engineering, Department of Robots and Production Systems, Bucharest, Romania

ABSTRACT

Mechatronics is a technology that has developed considerably over the years. A mechatronic system includes mechanical, electrical and electronic elements, controlled by a software program from a programmable logic controller (PLC). The controller can be of different types, starting from the fact that all the functions in the mechatronic system are available. The article focuses on the Arduino microcontroller, in which a robot (or a robot arm) can take the elements from a belt, thus performing an industrial process. The novelty elements are that with the development of technology, mechatronic elements must fulfill several functions, and an Arduino microcontroller program can take over many of these functionalities, easing the mechanical and electrical complexity of mechatronic systems.

KEYWORDS

Mechatronic System, Arduino PLC, Robot, Hardware Components, Software Code

1. INTRODUCTION

Mechatronics, a multidisciplinary branch of engineering, refers to the engineering of electrical, electronic, and mechanical systems. The name "mechatronics" has given for technical and practical considerations. The term originates from the Yaskawa Corporation from the combination of mechanics and electronics in 1969. After the 1970s, the meaning of mechatronic extended to include software and computation. [1]

With the advancement of technology, mechatronics development has diversified. The goal in mechatronics is to produce a designed solution that unifies all its previous components mentioned. However, the development of mechatronic systems has allowed multiple interactions between mechanical and electronic elements. Hardware and information-based functions software have allowed increasingly complex functions performed by a mechatronic system. [2] With the industrial revolution, hardware complexity has increased, and system crashes or malfunctions could occur. The initial controllers took over these functions, but with the newly implemented functions of the actuators and sensors, some functionalities were limited. Therefore, controllers developed to take over from these hardware functionalities in the software part, facilitating the mechanical part.

One such controller is the Arduino, which is small and robust, and can interconnect many elements and perform the functions desired by the system. [3] Many hardware elements such as an engine include encoders on it, and the functions operated are in greater numbers. Years ago, a mechanical part operated directly, regardless of sensors. With these built-in sensors, we have

more control, for example, but if controlling not done, the mechanical part only works partially correctly. [4]

2. THE MECHATRONIC SYSTEM WITH ARDUINO PLC AND ROBOT (ARM)

The system created took into consideration all the elements that define a mechatronic system. The designed product contains mechanical, electrical, and electronic elements. The resulting small-scale system reproduces the production process in an industrial environment. [5], [6]

The first phases of the realization of a mechatronic system requires a design. A designer should research the desired functions and choose hardware components that fulfil these functions required by the system. [7]

In the following phase, the implemented software desired of the code needs to execute exactly with the hardware components, so that there is no discrepancy between the functions performed by each component. Wide range of functions work through a specific code. This software is small and easy to handle, as one can easily make changes in the program. [8], [9]

2.1. The Hardware Components

The hardware components described below. The mechatronic system can identify the main functions explained. These hardware components permit the mechatronic system to perform the functions required. [10]

- ✓ The PLC module is an Arduino UNO board that allows the insertion of a programming code. The inserted code is used to control the treadmill as well as the inputs on the Arduino board (2 and 10), resulting in an output signal (on pin 13).

Arduino board has the following features, as in Table 1.

Table 1. Arduino board with features

| | |
|-----------------------------|--|
| Microcontroller | ATmega328P |
| Operating voltage | 5V |
| Input voltage (recommended) | 7-12V |
| Input voltage (limit) | 6-20V |
| Digital I / O | 14 (of which 6 provide PWM output) Pulse width modulation (PWM) is a technique for obtaining analogue results with digital means. Digital control used creates a square wave, a signal on and off. This on-off model can use voltages between the V in DC of the board (5V) and off (0 Volts) by changing, the time that the signal spends compared to the time spent by the signal. The duration "on time" called the pulse width. To obtain variable analogue values, the pulse width is changed or modulated. If is start-stop pattern repeats fast enough with an LED, for example, the result is as if the signal is a constant voltage between 0 and V in DC that controls the brightness of the LED. |
| PWM digital I / O | 6 |
| Analog input pins | 6 |
| DC current on I / O pin | 20 mA |
| Direct current for | 50 mA |

| | |
|---------------------|--|
| 3.3V pin | |
| Flash memory | 32 kB (ATmega328P) of which 0.5 kB used by bootloader (initial program load) |
| SRAM | 2 kB (ATmega328P) |
| EEPROM | 1 kB (ATmega328P) |
| Clock speed (CLOCK) | 16 MHz A crystal oscillator is an electronic oscillator circuit that uses the mechanical resonance of a vibrating piezoelectric crystal to create a constant frequency electrical signal. This frequency is often used to keep track of time, as in quartz watches, to provide a stable clock signal for digital integrated circuits, and to stabilize frequencies for radio transmitters and receivers. The 16 MHz crystal oscillator module is designed to handle off-chip crystals that have a frequency of 4-16 MHz The oscillator design generates low frequency and phase fluctuations, recommended for USB operation. |
| Integrated LED | 13 |
| Length | 68,6 mm |
| Width | 53,4 mm |
| Weight | 25 g |

The Arduino board containing the microcontroller is the one from figure 1.

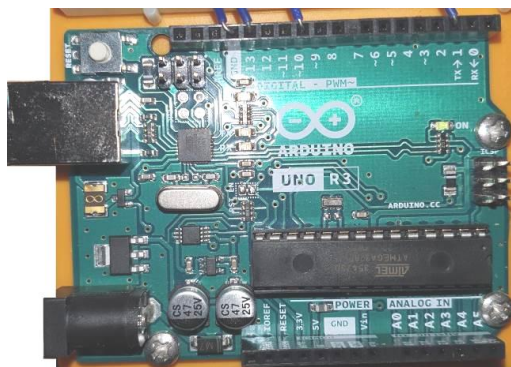


Figure 1. Arduino UNO type microcontroller

✓ Electronic components

- Two motion sensors, type HC-SR505, used as inputs for the mechatronic system; HC-SR505 is a mini PIR (Passive Infra-Red) motion sensor, used to detect the movement of a body in front of it. The device works based on infrared technology and can automatically control itself with high sensitivity and high reliability. Due to its minimal size and low power consumption, it is widely used in battery-powered applications.
- Four LEDs (light emitting diodes), a type of diode that illuminates when electricity passes through it. As with all diodes, it allows electricity to pass in one direction only. The anode (longer leg) connected to the source, while the cathode is the shorter leg, connected to the ground (GND). When the current is too high, the LED inserted with a resistance to GND to avoid burning; Resistors act on the flow of electricity in a circuit, changing the voltage and current. Resistor values measured in ohms, indicated by the colored stripes on the sides.
- Potentiometer - variable resistor with three pins. Two of the pins connected with the ends of a fixed resistor. The middle pin moves over the resistor, splitting it into two halves. When the outer parts of the potentiometer connected to the

voltage and ground, the middle pin will give the voltage difference when the knob is turned. The potentiometer is a simple button that offers a variable resistance, read in the Arduino board as an analogue value. In the system created, that value controls the rate at which an LED (yellow) lights up.

✓ Connections

- USB cable - allows the connection of Arduino with the computer for programming, which is the method of loading code into Arduino. This also provides Arduino power for many projects as the communication made on a port to activate the incoming and outgoing inputs.

The word "port" simply means a two-way portal or access. The combinations already set called COM ports for serial devices and LPT ports for parallel devices.

- Wires for sensors and power supply (plus and minus); as not all electronic components are on the connection panel, wires have been used for the sensor connections as well as the treadmill.
- The power supply of the Arduino did not allow the control of the mechatronic system, and it has added a variable voltage source. Here, is not used a battery because consumes from energy. This does not allow continuous control of the system, especially of the actuator.

A polarized diode was used at the minus of the power supply (and the current can only flow in the direction of the strip drawn on the diode), to allow electricity to flow in only one direction. This situation exists in a system that has a motor (like the current system) or high currents in the loaded circuit.

- ✓ The connection panel - very low currents acting through the system are an advantage, in the sense that is used a panel where the wires are inserted, as well as some electronic components.
- ✓ Treadmill - a motion (light) band modified to support two objects passing through the sensors.
- ✓ A robot with microcontroller and five axes, which takes the object from the position. The robot stores the object detected by the sensor in another position. The robot performs physical hard work, lifting the object from the movement belt. Robot arms are widely used in industry.

2.2. The Code from the Arduino Microcontroller

The data acquisition board controls the mechatronic system. The mechanical and electronic components execute the received commands. The software is an Arduino code.

The program written in Arduino:

```
int sensor1 = 2;           // pin corresponding to sensor 1
int sensor2 = 10;         // pin corresponding to sensor 2
int led = 13;             // pin corresponding to the led
int cond = LOW;           // by default, no motion is detected
int status1 = 0;          // sensor condition storage variable 1
int status2 = 0;          // variable for storing sensor condition 2
void setup () {           // configuration code, to run repeatedly
  pinMode (led, OUTPUT); // the LED is initialized
```

```

pinMode (sensor1, INPUT);           // initialize the sensor
pinMode (sensor2, INPUT);           // initialize the sensor
Serial.begin (9600);                 // the series is initialized
void loop () {                       // configuration code, which runs only once
  status1 = digitalRead (sensor1);   // read the value of the sensor 1
  status2 = digitalRead (sensor2);   // read the value of the sensor 2
  if (status1 == HIGH || status2 == HIGH) { // if one of the sensors is HIGH
    digitalWrite (led, HIGH);        // the LED lights up
    delay (100);                      // wait 100 milliseconds
    if (cond == LOW) {
      Serial.println ("Object detected near the sensor. Lifting object");
      cond = HIGH; } }               // the variable condition is updated with HIGH
  else // otherwise {
    digitalWrite (led, LOW);         // the LED goes out
    delay (200);                      // wait 200 milliseconds
    if (cond == HIGH) {              // updates the variable condition with HIGH
      Serial.println ("No motion detected");
      cond = LOW; } }               // update the variable condition with LOW
    delay (1000); } }               // wait, then resume execution

```

After the hardware part is completed, is integrated the software program.

3. THE DESCRIPTION OF THE SYSTEM

The Arduino controls the mechatronic system, whereas the mechanical and electronic components execute the received commands. The software written in Arduino code reflect all the functionalities desired.

The assembly of the components with the panel shows the part of the system in the figure below.

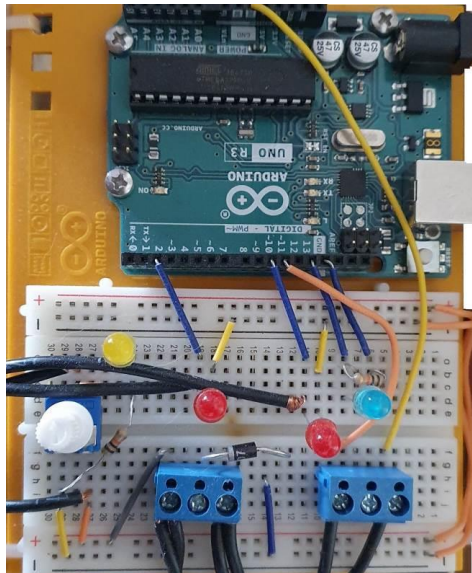


Figure 2. Arduino and connection panel

The microcontroller is communicating to the computer via the USB cable. From Arduino, the connections start, meeting on the panel. The panel enables connection with the treadmill, where the objects detected by the sensors move.

From the Arduino program, it is possible to track the status by displayed messages, with a serial rate of 9600 baud. Thus, one can observe the states detected at a given time.

System operation:

- when the system is functional, we turn on the yellow LED (for motion control), and the two red LEDs show the status of the sensors that encounter objects on the tape, and the blue LED is the output of the object detection;
 - the reported events via the serial connection (figure 3), where is displayed the detection or non-detection of the object;
 - the treadmill (controlled by the potentiometer on the panel), and the object are moving in front of a sensor (inputs 2 or 10 on the Arduino board); when detected, the blue LED is on, and a detection message appears in the serial connection window and the object can be lifted (by the robot or manually);
 - the operation performed can be continuous if there are powered components from the system.
- The whole system is in the figure below.

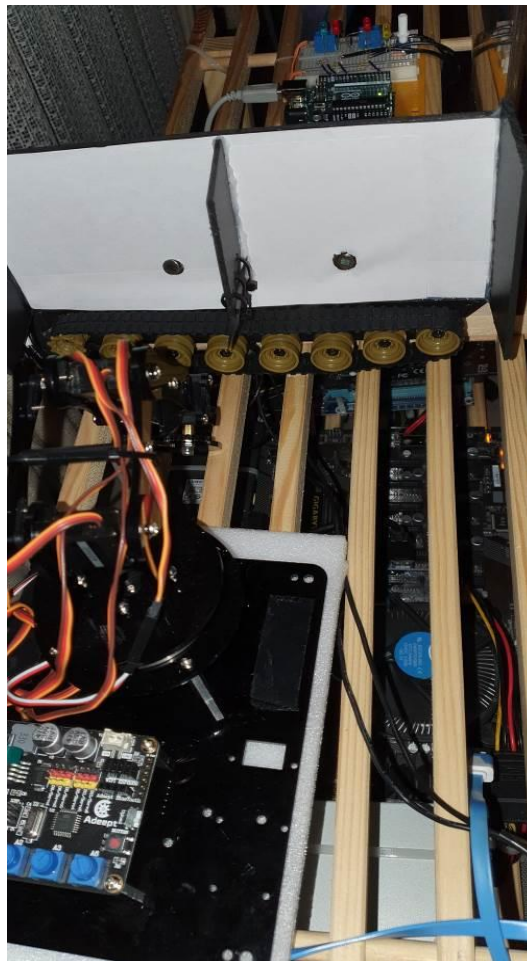





Figure 3. The created mechatronic system

The description of the system is in Table 2.

Table 2. Mechatronic system description

| | | |
|--|---|---|
| YELLOW LED ON |  | The treadmill is functional and not moving |
| YELLOW LED OFF | | The treadmill is moving |
| RED LED ON |  | Motion is detected at one of the sensors; the object can be lifted (inputs 2 or 10 from the Arduino controller) |
| RED LED OFF | | No movement is detected by any sensor (1 or 2) |
| BLUE LED ON |  | After a motion is detected by a sensor, the message for movement is displayed and the object can be lifted (exit 13 from Arduino) |
| BLUE LED OFF | | No motion is detected by any sensor |
| Note: the system has a variable DC power source (to control the potentiometer-driven treadmill and provide the required current surplus). It is power from the Arduino USB cable (to control the serial display and the exit from the blue LED). | | |
| Normal system status Yellow LED on (treadmill is working) RED LEDs are off (no motion is detected) Blue LED is off (motion not detected) | | |

In the built mechatronic system, an Arduino controller chosen to show what a mechatronic system is like and works with such a controller. Table 2 describes the operation of the system. It belongs to a mechatronic system, with all its functions.

4. CONCLUSIONS

With the help of a mechatronic system, many industrial processes perform accurate. The choice of hardware components that perform the functions required by the system studied as a priority before starting the construction of the system. The software component must be able properly controlling the hardware. The control system can cost quite a bit and can limit the system to certain features. Arduino PLC allows a very wide operating range, is not pretentious and is easy to control or edit the source code as well as uploading. The robot arm has a separate loaded code. It is located inside the module that controls the robot arm.

A client wants created a mechatronic system that detects objects on a moving tape. A robot can pick up these objects and leaves them in a certain position. The whole process monitors at this stage. Depending on the status of the system, a message reported or an LED commanded. This type of system is a classic, industrial type.

The novelty in this article is the integration of the Arduino controller in the mechatronic system. This microcontroller can perform all the desired functions at a low cost. One downside is the Arduino code, which requires programming knowledge. For other types of controllers, logic schemes used, which makes it easier to implement the desired functionalities. The future will reserve controllers that will require more programming elements, and Arduino is one of the good choices.

Over time, it is very possible that mechatronic systems will be more and more complex in terms of information technology (real-time computing). Software can “store” several hardware states (as is possible with a programmable logic controller), and the information provided to users can

be much more complex. The provided information via TCP / IP or Wi-Fi, and the price is currently high. Perhaps the hardware structure will remain approximately the same, at least in terms of functionality. From the point of view of detection (e.g. status), with the development of technologies, they can detect more and more states (intermediate, for example). This would help us to have a better control over a mechatronic system.

The information will be able to be transmitted wirelessly more and more and in greater quantity, facilitating the exchange of information between the systems. Until then, the elements of the mechatronic system more interconnected, are following the trend of changes in recent years. We will no longer be able to talk about mechanics, electrical, but about combinations of elements. We will no longer have hardware elements of a certain type, but combined actuators and sensors, such as an encoder motor, plus other implemented sensors. Probably, in 15-20 years we will have very complex mechatronic systems, where the hardware and software interfaces will be able to detect several states of product.

REFERENCES

- [1] Carryer J, Ohline R, Kenny T, (2011) *Introduction to Mechatronic Design*, Prentice Hall, Boston, ISBN: 978-0-1314335-6-4;
- [2] Alciatore DG, (2012) *Introduction to Mechatronics and Measurement Systems*, 4th ed., Department of Mechanical Engineering, Colorado State University, ISBN 978-0-0733802-3-0; pp. 1-20, 431;
- [3] Blum J, (2019) *Exploring Arduino: Tools and Techniques for Engineering Wizardry*, 2nd Edition, Wiley, New Jersey, USA, ISBN 978-1-1194053-7-5, Ch. 2;
- [4] Ballas RG, Pfeiffer G, Werthschützky R, (2009) *Elektromechanische Systeme der Mikrotechnik und Mechatronik, 2. Auflage*, Springer, Berlin/Heidelberg, Germany, ISBN 978-3-540-89320-2, Ch. 1, 3, 4;
- [5] Bolton W, (2019) *Mechatronics: Electronic Control Systems in Mechanical and Electrical Engineering*, 7th Edition, Pearson, New York, USA, ISBN 978-1-2922-5097-7, Ch. 1-4, 7-9;
- [6] Birglen L, (2018) *Mécatronique – 2e édition*, Dunod, Malakoff, France, ISBN 978-2-1007-7652-8, Ch. 1-3;
- [7] Isermann R, (2005) *Mechatronic Systems Fundamentals*, Springer-Verlag London, UK, ISBN 978-1-85233-930-2, Ch. 1, 4, 5;
- [8] Heimann B, Gerth W, Popp K, (2001) *Mechatronik (Mechatronics)* (Fachbuchverlag Leipzig), in German, ISBN 978-3-4462171-1-9, pp. 13-27, 71-86;
- [9] Monk S, (2016) *Programming Arduino: Getting Started with Sketches, Second Edition (Tab)*, 2nd Edition, McGraw-Hill, New York, USA, ISBN 978-1-2596-4163-3, Ch. 1, 2, 6, 7;
- [10] Sima LM, Zapciu M, (2020) *Connections and Interfaces of Mechatronic Components on Digital Factory*, Conference Proceedings of the Academy of Romanian Scientists, PRODUCTICA Scientific Session, Volume 12, Number 1/2020, Bucharest, Romania, ISSN (print) 2067-2160, (online) 2067-9564, pp. 129-140.

AUTHOR

Liviu Mihai SIMA was born in 1979 and obtained Bachelor in Engineering in 2002 from University “Politehnica” Bucharest, Romania. He furthered his academic career, completing a Master’s degree, and is currently working towards his Ph.D. at the same University. He has published various articles in the field of transportation, mechatronics, etc. His field interests include computer networks, IoT, railway transportation, and mechatronics.

