sciendo

Műszaki Tudományos Közlemények vol. 13. (2020) 45–49.
DOI: Hungarian: https://doi.org/10.33895/mtk-2020.13.05
English: https://doi.org/10.33894/mtk-2020.13.05

EME MTSZ MTK

RECONSTRUCTION OF A GRINDING MACHINE INTENDED FOR THERMO-POLYMERS, AND DESIGN OF A LINUX BASED IOT CONTROL SYSTEM

Attila DEBRECENI,[1] Timotei István ERDEI,[2] Szabolcs TÓTH,[3] Géza HUSI[4]

University of Debrecen, Faculty of Engineering, Debrecen, Hungary

[1] *adebreceni93@gmail.com*

[2] *timoteierdei@eng.unideb.hu*

[3] *szabolcs978@gmail.com*

[4] *husigeza@eng.unideb.hu*

Abstract

With the increasing use of 3D printing technology, a closely related problem is that of spreading. This problem is the presence of the polymer waste created by faulty prints, and support material used during printing. To create filament from this waste, it must first be chopped into fine pieces. In this project, an original polymer grinder was designed and built, adopting the innovations of Industry 4.0. Remote control and supervision were achieved using a Raspberry Pi I. type B and an Arduino Nano. The finished project can be seen in the faculty of engineering at the University of Debrecen.

Keywords*: SketchUp make, 3D modelling, gripper, robot.*

1. Introduction

Various industry 4.0 innovations have a significant impact on the industries of the 21st century. One such innovation is the emergence of remote control and remote supervision systems which mainly use the internet as a communicational channel.

Using reliable technologies, a remote control and supervision system was designed, suitable for industrial use in the Cyber-Physical & Intelligent Robot Systems Laboratory of University of Debrecen, Department of Mechatronics [1].

2. Choosing a suitable technology

An FDM 3D printer was previously built in the Cyber-Physical & Intelligent Robot Systems Laboratory, which, during regular usage, produced a considerable amount of plastic waste. Most of this waste is so called “support” material, however, faulty prints also contribute to this due to the substantial amount of material used during printing. These faults can be traced back to multiple issues. One such issue is that models printed from ABS have a tendency whereby, in relatively tall models, the layers crack, or separate. Other examples are incorrect extruder feed rates (either too fast or too slow), or the use of incorrect temperature ranges for the extruder, or heat bed.

With the appearance of this waste, designing a recycling system was considered. Recycling can be separated into two processes. First is the grinding of waste, followed by the creation of filament from the ground waste. This project focuses on the process of grinding.

The first task was to choose a suitable grinding method. A hammer mill was chosen [2] for the following reasons:

- The machine was originally intended for materials of similar tensile strength.
- The hammers have good parts supply.
- The motor can be swapped if needed.
- The machine is easy to repair, maintain, and expand.

3. Reconstruction of the mill

In the grinding chamber, the material to be ground is broken into pieces by the high RPM hammers, breaker plate, and finally, enters the sieve. The sieve ensures that only pieces smaller than a specific diameter can pass through. After choosing this technology and acquiring a mill, the machine had to be restored. As it was earlier used for grinding grains (such as wheat), it was imperative to thoroughly clean the machine to make it suitable for processing plastics. The reconstructed machine is shown on **figure 1**.

A suitable source of mechanical power also needed to be selected for the mill. From the available options, an Agisys MS 803-4 **[3]** three phase, asynchronous electric motor was used, capable of 1.1 kW of power at 1390 [RPM] when operated from a 400 V / 50 Hz power source.

Figure 1. *Reconstructed mill.*

4. Circuit design

The following devices were chosen for the circuit:

a Raspberry Pi, an Arduino Nano, a contactor, a TRIAC, an opto-isolator, the previously mentioned three phase AC motor, and an MPU6050 accelerometer.

The circuit was designed in KiCAD. It can be observed on the circuit how the motor is controlled by the Arduino Nano **[4]** which is connected to a Raspberry Pi 1 **[5]** via USB. These two devices exchange data using serial communication. This connection makes it possible to control the mill remotely, and to access various supervising functions. Output 7 of the Arduino switches the LED of the MOC3062 opto-isolator, which then allows mains power to reach the TRIAC's gate. Then, the BTA208X TRIAC's main terminals are able to supply the control voltage required by the contactor, which then uses the 230 V control voltage to switch the three phase 400 V voltage and operate the electric motor. These connections can be seen on **figure 2.**

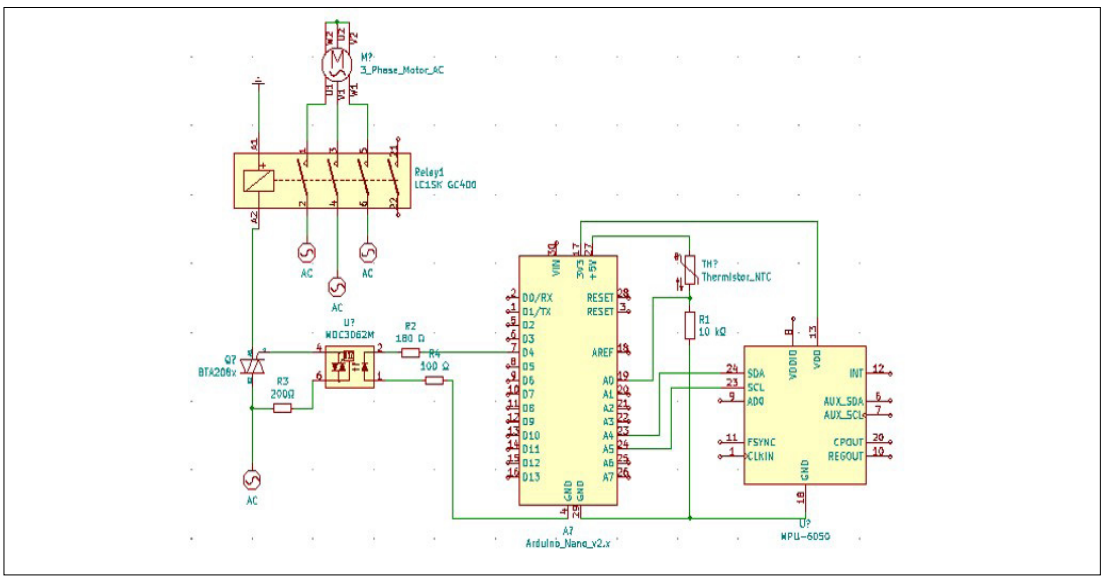

Figure 2. *The motor controller circuit.*

An NTC thermistor was added to the motor in a voltage-divider configuration. Using the thermistor's characteristics (with a suitable program code), it is possible to tell if the motor is overheating or not. This information is also displayed on the web page. The MPU6050 accelerometer can be used to conduct vibration analysis, to monitor the mill's mechanical state.

5. Preparations for programming, achieving remote control

During programming, five different programming languages were used to achieve the controlling logic seen on the flowchart from **Figure 3**.

These programming languages are as follows. PHP, HTML, a JavaScript library, Python 3, and C++. PHP and Python 3 were written in the Raspberry's terminal text editor (GNU Nano **[6]** while the C++ code was written using the Arduino IDE.

Before programming, various configuration steps had to be completed on the Raspberry Pi. On this device, the Raspbian Lite **[7]** version was installed. Apache2 **[8]** is an open source http webserver. To ensure that the Arduino was able to communicate with the php program, the PhPSerial library had to be downloaded. The MPU6050 communicates with the Arduino via I2C, where the Arduino is the master, and the MPU6050 is the slave device. I2C communication is implemented using Arduino's wire.h library. The first function was remote access. The program responsible for this on the Raspberry Pi is shown in **Figure 4**.

First the PhPSerial was added to the program, then the communication port and baud rate was set up, which was used to set up connection with the Arduino. The deviceOpen() command starts communication. The webpage has two buttons, which send a "Start" or "Stop" command.

In the case of a Start command, a message is sent via the serial port to the Arduino, which contains the number "6", and writes the contents of the $msg variable to the "Az aprító üzemel" (the mill is running) message, which shows up on the webpage. In the case of a stop command, the number "7" is sent, and the message on the webpage changes to "Az aprító leállítva" (the mill is stopped).

In the Arduino's void loop segment, if any information arrives via the serial port, it is immediately associated with the "incoming" state variable. If "6" arrives, the pin is turned into a "HIGH" state, which causes the control voltage to be routed to the contactor's inductor via the TRIAC, thus

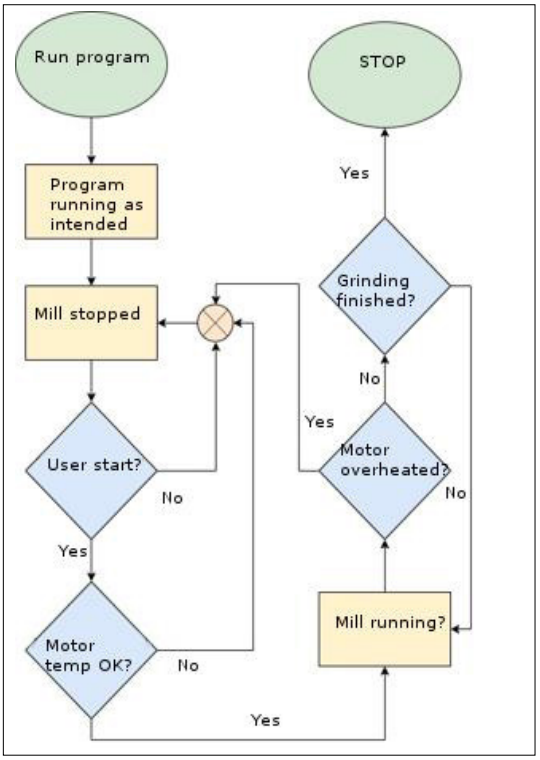

Figure 3. *Flowchart of grinding.*

```
include "phpSerial.php";
$comPort="/dev/ttyUSB0";
$msg ='';

    $serial = new phpSerial;
    $serial->deviceSet($comPort);
    $serial->confBaudRate(9600);
    $serial->deviceOpen();
    sleep(2);

if(isset($_POST['command'])){
    if($_POST['command'] == "start"){
    $serial->sendMessage("6");
    $msg = "Az aprító üzemel";
    }

    if($_POST['command'] == "stop"){
    $serial->sendMessage("7");

    $msg = "Az aprító leállítva";

}
    $serial->deviceClose();
}
```

Figure 4. *Part of the code responsible for remote access.*

supplying the AC motor with electric power, as long as it's not overheated.

In the index file a slider was also created, which can take up a value ranging from one to five, in steps of one. These values are sent as "MotValue", and are displayed on the website as text in an input window. The "MotValue" value determines the amount of time the motor will be turned on for. In the Arduino code, the millis function is used, where the current millis value is stored in a variable, after which relative time measurement can take place.

6. Utilising the thermistor and MPU6050

As protection, the motor is equipped with an NTC MF52B2 thermistor [9]. This thermistor can return usable values in a -50 to +125 °C range.

As seen on figure 5, the thermistor has a 100 [kΩ] resistance at room temperature. In the voltage divider, a 10 kΩ resistor can also be found. This solution ensures that the analog 0 pin can be used to read the voltage value as temperature. A 5 V voltage source was connected to the voltage divider. As such, the Arduino automatically perceives this input as a 0–1023 range. Using an equation, the program is then able to calculate the resistance of the thermistor. Later, the temperature is calculated using a Steinhart equation, which is also found in the program.

Afterwards, in the void loop segment, the accelerometer's values are recorded. The MPU 6050 uses a MEMS sensor, and returns the measured acceleration in a mg/LSB format. The "mg" value is the gravitational acceleration with a "milli" prefix, while LSB stands for least significant bit. The sensor can be configured for various acceleration ranges, which are: ±2g, ±4g, ±8g, ±16g. In this case, ±8g was chosen, which means that the total value range is 16000 [mg]. This is stored in 16 bit, so a total of 216 values is possible. The measured voltage value and the accelerometer's data are then both sent to the Raspberry from the Arduino via serial communication.

On the Raspberry, a python 3 script was written as seen on figure 6, to record the measurement data of the thermistor and the accelerometer. This script is started as soon as the Raspberry is turned on.

The program first creates a data list, which splits the data coming from the serial port into 4 separate groups. This is necessary to prevent the acceleration and temperature data from mixing. These groups are created based on when said

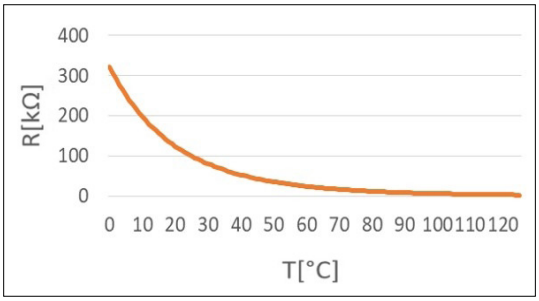

Figure 5. *NTC thermistor temp-resistance graph.* [9]

```
import serial

ser=serial.Serial('/dev/ttyUSB0',9600)
numPoints = 4
dataList = [0]*numPoints
temp_file = open("/home/pi/Desktop/temp/temp_data.txt",'wb')
accX_file = open("/home/pi/Desktop/temp/accX_data.txt",'wb')
accY_file = open("/home/pi/Desktop/temp/accY_data.txt",'wb')
accZ_file = open("/home/pi/Desktop/temp/accZ_data.txt",'wb')
def getValues():
    sensor_data = ser.readline()
    return sensor_data
while ser.read():
    for i in range(0,numPoints):
        data0 = getValues()
        data1 = getValues()
        data2 = getValues()
        data3 = getValues()
        data = getValues()
        dataList[i] = data
        dataList[0] = data0
        dataList[1] = data1
        dataList[2] = data2
        dataList[3] = data3
        print(data0)
        print(data1)
        print(data2)
        print(data3)
```

Figure 6. *Python program handling sensor data.*

data arrives after the previous, so to define the separate measurements, a "getValues()" function was defined. This saves the values arriving from the serial communication as "sensor_data" using the ser.readline command. This always reads all incoming bytes, then a for cycle is started whenever a message arrives. Within this for cycle 5 variables are created: 4 numbered "data" variables, and one non-numbered "data" variable, which target the "getValues()" function. The created 4 item data list is associated with the 4 numbered data variables, and the for loop reads the incoming packets one by one. This ensures that every variable only contains the variable that corresponds to that number in the order they arrive in.

The non-numbered data variable contains the entire list, which can be requested at any time.

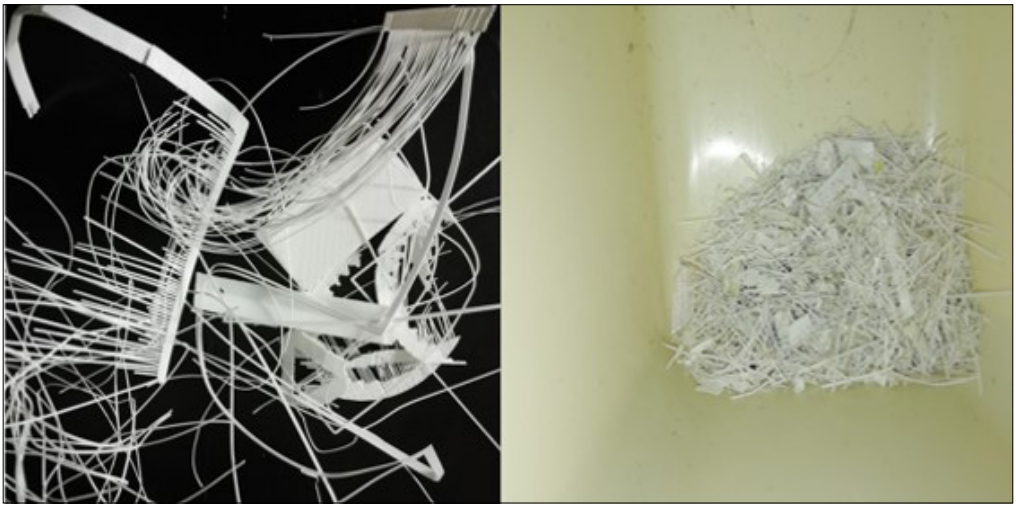

Figure 7. *Grinding results.*

This can be used for troubleshooting purposes, especially when the incoming data is mixed up due to an error.

The separated data then needs to be saved into text files, which is achieved by the ".write()" and ".flush()" commands. The flush command is needed to also flush the data from the buffer directly into the file, which can prevent potential data losses, compared to only using the ".write()" command.

A php script and a JavaScript code was added to the original webpage's HTML code. With this, it's possible to show periodically refreshed data on the webpage, without reloading the webpage itself.

7. Grinding industrial ABS

After these programming tasks and circuit assembly was completed, a test run was conducted. The test object was completely milled after 2.16 minutes, the final result is shown on **figure 7**. The result matches the previously set requirements.

8. Conclusion

During the project, the previously set goals were achieved, and thoroughly documented. The mill can be operated remotely, and a supervising system was also added. At the end, a test was conducted, which showed satisfactory results.

Acknowledgements

I would like to thank Géza Husi's for his help, and who assisted me whenever questions arose. I'd also like to thank the Faculty of Engineering of University of Debrecen for providing everything that was necessary for the completion of this project. This research was supported by the Doctoral School of Information Technology, University of Debrecen.

References

[1] T. I. Erdei, Zs. Molnár, N. C. Obinna, G. Husi: *Cyber physical systems in mechatronic research centre*. MATEC Web Conf. Volume 126, 2017.

[2] Kacz K.: Aprítógépek (darálók) felépítése, működése. In: Állattartás műszaki ismeretei. (2011) (2019.11.20). https://www.tankonyvtar.hu/hu/tartalom/tamop425/0010_1A_Book_14_Az_allattartasi_muszaki_ismeretek/ch04s02.html

[3] Agisys MS 803-4 (2019.11.21) https://agisys.hu/up/docs/agisys-motor-katalogus.pdf

[4] Arduino Nano (2019.11.21) https://www.arduino.cc/en/Guide/ArduinoNano

[5] Raspberry Pi 1 Model B+ (2019.11.23) https://malnapc.hu/raspberry-pi-1-model-b

[6] GNU Nano (2019.10.20) https://www.nano-editor.org/

[7] Raspbian Lite (2019.11.23) https://www.raspberrypi.org/downloads/raspbian/

[8] Apache 2 HTTP Server (2019.11.23) https://httpd.apache.org/

[9] Specifications for NTC Thermistor (2019.11.25) https://www.tme.com/Document/f9d2f5e38227fc1c7d979e546ff51768/NTCM-100K-B3950.pdf.