sciendo

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

EME MTSZ MTK

TERMOPOLIMER DARÁLÓEGYSÉG REKONSTRUKCIÓJA ÉS IOT-IRÁNYÍTÁSTECHNIKAI RENDSZER TERVEZÉSE LINUX-ALAPON

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

Debreceni Attila,[1] Erdei Timotei István,[2] Tóth Szabolcs,[3] Husi Géza[4]

Debreceni Egyetem, Műszaki Kar, Debrecen, Magyarország

[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.*

Összefoglalás

A 3D-nyomtatás terjedésével párhuzamosan terjed egy probléma, amely a technológiával szoros összefüggésben áll. A probléma nem más, mint a nagy mennyiségű polimer hulladék, amelyek forrása többek között a sérült modellek és a support-anyagok. Ahhoz, hogy a hulladékból szálat lehessen húzni, elsőnek apróbb darabokra kell őket aprítani. A projektben egy egyedi kivitelezésű polimeraprító lett megtervezve és megépítve, amely adaptálja magába a 4. ipari forradalom követelményeit. A távvezérlést és a távfelügyeletet egy Raspberry Pi I. type B és egy Arduino Nano segítségével került megvalósításra. Az elkészült projekt megtalálható a Debreceni Egyetem Műszaki Kar Mechatronika Tanszékén.

Kulcsszavak: *ABS, távfelügyelet, távvezérlés, Raspberry Pi, Arduino, Linux, IoT, Industry 4.0.*

1. Bevezetés

A XXI. század iparát jelentős mértékben befolyásolja az Ipar 4.0 vívmányai. Ezen vívmányok közé tartozik a távvezérlési és a távfelügyeleti rendszerek, melyek legfőbb csatornája az internet.

Az üzembiztos technológiák alkalmazásával kiforrott, az ipar számára is használható távvezérlés és távfelügyeleti rendszert került kidolgozásra, a Debreceni Egyetem Mechatronikai Tanszék, Cyber-Physical & Intelligent Robot Systems Laboratory-ban [1].

2. A megfelelő technológia kiválasztása

A Cyber-Physical & Intelligent Robot Systems Laboratoryban korábban megépítésre került FDM 3D-nyomtató, rendszeres használata során jelentős mennyiségű hulladék keletkezik. Ezek jelentős része a „support”-támaszelemek, de nem elhanyagolható a hibás modellek száma sem, melyek jelentős mennyiségű polimerből épülnek fel. Ezek a meghibásodások több tényezőre is visszavezethetőek, mint például az a tény, hogy az ABS-rétegekből felépített modelleknél gyakori jelenség, hogy jelentős magasságok esetén előfordul a rétegek elválása-repedése. Problémát okozhat a nem megfelelő szálhúzásra vonatkozó beállítások (túl lassú vagy túl gyors). Illetve a heatbed és az extruder rossz hőmérséklet-tartomány megadása.

A hulladékok megjelenésével számításba került egy újrahasznosító rendszer kidolgozása. Az újrahasznosítás két részre bontható. Az első a hulladékok felaprítása, míg a második a már feldarabolt polimerből a szálhúzás. A projekt során az első rész megvalósítása élvezett prioritást.

Az első feladat a megfelelő aprítási technika kiválasztása volt. A választás egy kalapácsos aprítóra [2] esett a következő okokból:

- Az eszköz eredetileg hasonló szakítószilárdságú anyagok zúzására lett tervezve.
- A munkát végző kalapácsok beszerezhetők.
- A motor cserélhető.
- Gép könnyen javítható, bővíthető.

3. Az aprító működésének ismertetése és rekonstrukciója

A darálótérben a darabolandó anyag a nagy fordulatszámú kalapácsokhoz, majd törőfelülethez, végül a rostafelülethez ütközve aprózódnak fel. A rosta szerepe, hogy csak azok a darabok juthassanak ki a darálótérből, amelyek mérete kisebb, mint a rosta átmérője. Miután a technológia megválasztásra került és a darálóegység beszerzése megtörtént, szükséges volt annak felújítása. Mivel az eszközt korábban már alkalmazták termény darálására és a célból, hogy megfeleljünk a tisztasági követelményeknek a ledarált termopolimer esetén elvégeztük a kötelező tisztítási folyamatokat. A felújított berendezés az 1. ábrán látható.

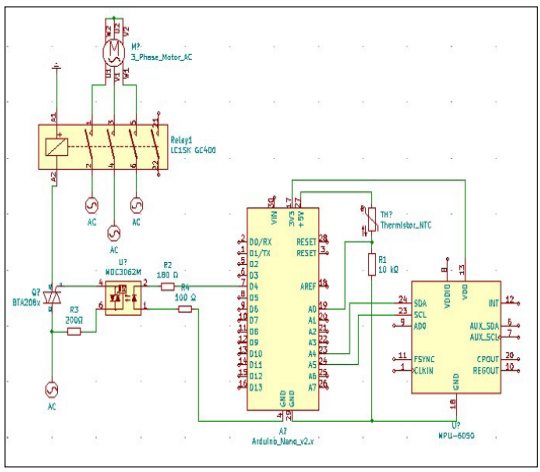
A polimerdarálóhoz ki kellett választani a megfelelő meghajtást is. A rendelkezésre álló eszközök közül egy Agisys MS 803-4 típusú 3 fázisú 400 [V] feszültségen működő, 1,1 [kW] teljesítményű, 50 [Hz] hálózati frekvencián 1390 min^{-1} fordulatszámú aszinkron motort [3] lett választva.

4. A megépített áramkör

Az áramkörhöz a következő eszközök kerültek kiválasztásra: Raspberry Pi 1 Arduino Nano, 1 kontaktor, 1 TRIAC, 1 optoizolátor, 3 fázisú AC motor, MPU6050 giroszkóp és gyorsulásmérő szenzor.

A 2. ábrán látható áramkör KiCAD-ben került megtervezésre. A kapcsoláson látható hogy a motor vezérlését egy Arduino Nano [4] végzi, amely egy Raspberry Pi 1-re [5] van kötve USB-porton keresztül. A két eszköz soros kommunikáción keresztül küld és kap adatot egymástól. Ezen

1. ábra. *A rekonstruált darálóegység*



2. ábra. *A motort vezérlő elektronika*

kommunikációk révén a jelenlegi gépet lehetséges távolról irányítani, illetve a megvalósított felügyeleti rendszerek esetrén is ezen a porton kommunikál a 2 eszköz. Az Arduino 7-es kimenete MOC3062-jelzésű optoizolátor LED-jét bekapcsolja ezáltal az integrált áramkörön keresztül folyhat a hálózati áram, amely egy TRIAC gate bemenetére van kötve, így a kimenetét a BTA208X TRIAC gate csatlakozójára kötve biztosítja, hogy a „main” termináljain keresztül biztosítani lehessen a kontaktornak a szükséges vezérlőfeszültséget, melyen keresztül a motor megkapja a 3 fázisú 400 V feszültséget közvetlenül a hálózatról.

A motorban található NTC termisztor segítségével lehetséges megállapítani, hogy a motor nem melegedett-e túl, a motor hőmérsékletének az állapotát megjeleníti a weblapon. A kapcsoláson látható még egy MPU6050 gyorsulásmérő és giroszkóp szenzor. A szenzort felhasználva képesek lehetséges rezgéselemzést végezni.

5. A programozás előkészítése és a távvezérlés megvalósítása

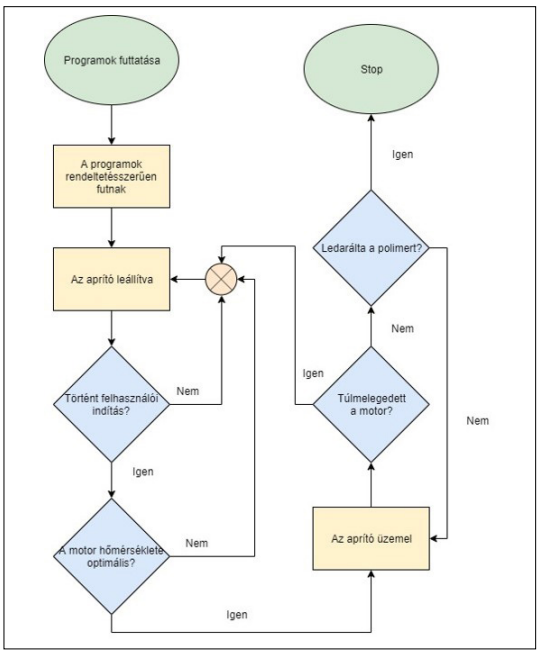
A programozás során öt különböző programnyelvet lett felhasználva, annak érdekében, hogy a lentebb látható folyamatábra kivitelezésre kerülhessen.

A programnyelvek közül PHP-t és a HTML-t, JavaScript könyvtárat, a Python 3, és C++ programnyelvek kerültek felhasználásra. A PHP és a Python 3 a Raspberryn futó terminál szövegszerkesztőjében a GNU Nano [6] környezetében lett megírva, míg a C++ kódot az Arduino IDE szoftverben.

Mielőtt a programozást elkezdődhetett volna, szükséges volt elvégezni a konfigurációs műveletet a Raspberry Pi eszközön. Az eszközre a Raspbian Lite [7] verzió került feltelepítésre. Az Apache 2 [8] amely egy nyílt forráskódú http webserver. Ahhoz, hogy a php-program kommunikálni tudjon az Arduinóval, szükségünk volt a PhpSerial nevű könyvtár letöltésére. Az MPU6050-szenzor I2C-kommunikációt folytat az Arduinóval, ahol az Arduino a master és az MPU6050 a slave eszköz, az I2C kommunikációt a wire.h library tartalmazza. A programok működésének áttekintését a 3. ábra segíti.

Az elsőnek megvalósított funkció a távoli elérés volt, melynek programja a 4. ábrán látható. Az ezt vezérlő program a Raspberry Pi-on a következő. Elsőnek hozzáadásra került a PhpSerial könyvtár a programhoz, majd megadásra került, hogy melyik porton és milyen baud rate-beállítással csatlakoztattuk az Arduinót. A deviceOpen() parancs indítja a kommunikációt. A weblap tartalmaz két 2 nyomógombot, ezek a nyomógombok küldenek egy „Start” vagy „Stop” parancsot command néven.

Start utasítás esetén soros porton keresztül egy üzenetet küld az Arduinónak, amely tartalma a



3. ábra. *Aprítás folyamatábrája*

```
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();
}
```

4. ábra. *A weblap távvezérlését irányító kód részlet*

„6” és a $msg változót tartalmát átírja „Az aprító üzemel” üzenetre amely megjelenik a weblapon. Stop utasítás esetén „7” üzenetet küld a soros porton, és a weblapon „Az aprító leállítva” üzenet jelenik meg.

Az Arduino kód Void Loop szakaszában, ha a soros porton információ érkezik az Arduinóhoz, azt az információt hozzárendeljük az incoming_state változóhoz, ha “6” érkezik a pint HIGH állapotba kerül, ami miatt a triacon keresztül a szükséges vezérlőáram eljut a nagy teljesítményű kontaktor tekercséhez, amely így húzni kezd, és így az AC-motor hálózatra lesz kötve, ha nincs túlmelegedve.

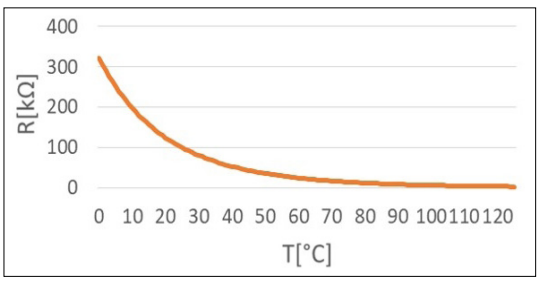
Az indexfájlban került szintén létrehozásra a csúszka, mely 1 és 5 között tud felvenni értékeket 1 egység lépésenként. A felvett értékek továbbításra kerül MotValue néven. A MotValue érték határozza meg, hogy hány percig legyen bekapcsolva a motor. Ez az Arduino kódban a millis függvény segítségével lett megvalósítva, a gombnyomáskor lévő millis érték rögzítésével, melyet követően relatív időmérés valósulhat meg.

6. A termisztor és a MPU 6050 kezelése

A motor védelme érdekében elhelyezésre került egy NTC MF52B2 termisztor **[9]**. Ezen termisztor -50 és +125 Celsius fok közötti tartományban képes a valóságra megközelítőleg igaz adatokkal szolgálni.

Ahogyan az **5. ábrán** is megfigyelhető, a termisztor szobahőmérsékleten 100 kΩ ellenállással rendelkezik. A feszültségosztó kapcsolásban található még egy 10 kΩ-os ellenállás, ezzel a megoldással az analóg 0 csatlakozót felhasználva, mérhető a motor pillanatnyi hőmérséklete.

A feszültségosztóra 5 V feszültség lett kapcsolva, így az Arduino a mért feszültséget automatikusan egy 0 – 1023 tartományra bontva értelmezi. Ezután a hőmérsékletet a szintén a programban látható Steinhart–Hart-egyenlettel számíthatjuk ki.



5. ábra. *NTC-termisztor ellenállás-hőmérséklet görbéje*

Lekérdezésre kerülnek a gyorsulásmérő adatai. Az MPU 6050 egy MEMS-szenzort használ, a mért gyorsulási adatokat mg/LSB formában adja meg. A szenzornak be lehet állítani a mérési tartomány nagyságát, melyek a következők: ±2g, ±4g, ±8g, ±16g. A jelen esetben a ±8g tartomány lett használva, ami azt jelenti, hogy összesen 16000 [mg] széles tartományban mér a szenzor, a kimenet 16 bites, így összesen 216 különböző értéket tud felvenni. A mért feszültségértéket és a gyorsulásmérő által mért adatokat az Arduino továbbítja soros porton a Raspberrynek.

A Raspberryn egy, a **6. ábrán** látható python 3 szkript került megírásra, amely rögzíti a termisztor és a gyorsulásmérő adatait. Ez a skript automatikus elindul, ha bekapcsoljuk a Raspberryt.

A program soros porton keresztül elsőnek létrehoz egy adatlistát, amely a beérkező adatokat 4 külön csoportba bontja. A csoportok elemeit aszerint választja ki, hogy hányadik sorban érkeznek, ezért arra, hogy a szenzorok mérése definiálásra kerüljön, kell egy get Values() funkció, amely sensor_data néven lementi a soros porton olvasott adatokat a ser.readline paranccsal. A létrehozott 4 elemű adatlista hozzá lett rendelve a 4 számozott data-változóhoz és a for loopnak köszönhetően egyesével lépkedve olvassák le a bejövő csomagokat. A számozatlan data változó a teljes listát

```
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)
```

6. ábra. *A szenzor adatokat kezelő python program*

7. ábra. *Aprítás eredménye*

tartalmazza, amit bármikor lekérdezve látható a teljes lista tartalma, ezzel lehet hibát keresni, ha az adatok összekeveredtek.

A külön változókba rendezett adatokat le kell menteni a létrehozott szöveges fájlokba. Ezt a funkciót a .write() és a .flush() paranccsal lehet megtenni. A flush parancsra azért van szükség, hogy a belső pufferből a fájlba közvetlenül ezáltal elkerülhető az adatvesztés, amely akkor történne, ha csak a.write() utasítást lenne használva.

Az eredeti weblap html-kódjához hozzáadva egy javascriptet, és egy php-kód megírásával képesek vagyunk arra, hogy a weblapon megjelenítsünk egy adatot, amely meghatározott időközönként frissül, úgy, hogy a weblapot nem kell újra tölteni.

7. Ipari ABS aprítása

A programozási feladatok és a kapcsolás megvalósítása után elvégezésre került egy aprítási teszt. A fent balra látható elemet 2,16 perc alatt aprította össze a gép. Az előre lefektetett minőségi feltételeknek pedig megfelelt a végeredmény. A végső eredmény a **7. ábrá**n látható.

8. Következtetések

A projekt során kitűzött célok teljesültek, és ezek a projektbeszámolóban kifejtésre kerültek. Az aprító rendelkezik távolról vezérléssel, a felügyeleti rendszer beépítése megtörtént, és az aprítási tesztet is sikeresen teljesítette.

Köszönetnyilvánítás

Ezúton szeretném megköszönni Husi Géza segítőkész hozzáállását, aki bármilyen probléma felmerülése esetén segítségemre sietett. Emellett megköszönöm a Debreceni Egyetem Műszaki Karának a szükséges feltételek biztosítását a projekt elkészültéhez. A kutatást a Debreceni Egyetem Informatikai Tudományok Doktori Iskola támogatta.

Szakirodalmi hivatkozások

[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.