Monitoring Print Engine Output Using Arduino and Raspberry Pi

1st Meiyanto Eko Sulistyo
Dept. of Electrical Engineering
Sebelas Maret University
Surakarta, Indonesia
mekosulistyo@staff.uns.ac.id

2nd Stephanus Hanurjaya
Dept. of Electrical Engineering
Sebelas Maret University
Surakarta, Indonesia
32.stephanushanurjaya@gmail.com

3rd Muhammad Danang Prastowo
IT Infrastructure Division
PT. Tiga Serangkai Inti Corpora
Surakarta, Indonesia
dprastowo@tigaserangkai.co.id

***Abstract*— In the printing industry process, monitoring is necessary for quality control of the product. The making of the tool on this project serves to monitor the output of the production machine. This monitoring is done by detecting the product output from the production machine using Sensor E18 D80NK. When the sensor detects the output, the sensor sends a signal to the Arduino UNO R3 which will calculate the amount of output from the product. Arduino will send information of the number of outputs via a USB connection to a central computer that is a Raspberry Pi 3 model B. The Python program on Raspberry Pi will read input from each Arduino address and display the data in realtime. At the same time, the data will be stored as a text file. This text file contains the number of product output and the time of the output. The prototype of this tool has been successfully created and there is still much development to do.**

Keywords*— *arduino, monitoring, python, raspberry pi

I. INTRODUCTION

In a printing production process, monitoring needs to be done in all aspects, from the pre-printing process, the printing process, the finishing, to the distribution process. On a printing machine, there is a screen to see how much input and output the process is working on. This reading is carried out by a sensor on the machine, and is displayed using a seven segment screen or an LCD display. The displayed figure will be manually checked by engine maintenance personnel. If there is a problem with the machine, this officer will handle the machine so that the production process is not interrupted.

There are so many printing machines operating simultaneously in a printing machine group that it is quite difficult to monitor the performance of all machines simultaneously. It takes a tool that can show the performance of each machine operating on a computer screen.

II. RESEARCH METHODS

A. *How the Production Machine Output Monitoring Tool Works*

PT. Tiga Serangkai Inti Corpora uses a variety of production machines. The working concept of the sensor output on each machine is similar. The sensor commonly used in production machines is the E18D80NK infrared sensor. The sensor reads the pause when the item passes through the sensor.

The Sensor E18 D80NK will read the engine output. This sensor is active LOW. When an item passes, the sensor will provide a LOW input which will be read by the Arduino UNO R3. The program in the Arduino UNO R3 will calculate the amount of output from the production machine. The results of the calculations carried out by the program on the Arduino UNO R3 will be displayed on the TM1637 LED display and also sent via a USB connection to the Raspberry Pi for further processing.

Raspberry Pi requires a program to read input from the Arduino UNO R3 which has been sent via USB. The program on the Raspberry Pi is made using the Python programming language. This program will read the input from Arduino and then display it in realtime GUI. The program will also save the displayed data in a text file. This text data will store information about the engine output and the time entered from the machine's output.

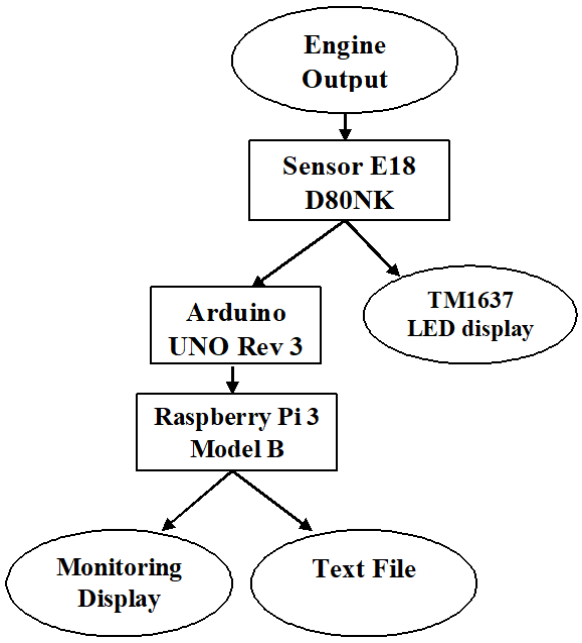


Figure 1 Block Diagram of Production Machinery Monitoring Tools

B. *Sensor E18 D80NK*

The Sensor E18 D80NK is an infrared sensor that has a far enough detection distance and little visible light interference. The implementation of the modulated IR signal makes this sensor immune to interference caused by normal light from a light bulb or from sunlight. This sensor has a detection distance that can be adjusted using a screwdriver. This sensor produces a digital output when it detects something within its detection distance. This sensor does not measure distance from objects. This sensor can be used to avoid collisions on robots and automated machines. This sensor provides non-contact detection. [1]

Figure 2. Sensor E18 D80NK [1]

Table 1. Specifications of the Sensor E18 D80NK [1]

Input Voltage	+5V DC
Current consumption	> 25mA (min) ~ 100mA (max
Dimension	1.7cm (diameter) x 4.5cm (length
Cable length	45cm
Detection of objects	Transparan atau buram
	Diffuse reflective type
Sensing range	3cm to 80cm (depends on obstacle surface)
	NPN output (normally high)
Environment temperature	-25 °C ~ 55 °C
Red wire	+5V
Green wire	GND
Yellow wire	DIGITAL OUTPUT

C. *Arduino UNO Rev 3*

Arduino Uno is a microcontroller board based on the ATmega328P. The Arduino has 14 digital input / output pins (6 of these pins can be used as PWM outputs), 6 analog inputs, 16 MHz quartz crystals, a USB connection, a power jack, an ICSP header and a reset button. The Arduino has everything you need to support a microcontroller; Connect the Arduino with a USB cable, AC-DC adapter, or battery to run Arduino. "Uno" is taken from Italian which means "first". The word "Uno" was chosen to commemorate the release of arduino software (IDE) 1.0. Arduino Uno and version 1.0 arduino software (IDE) are the version numbers of Arduino, currently a newer version has been developed. The Uno board is the first in the series of USB Arduino boards, and the reference mode for the Arduino platform. [2]

Figure 3. Arduino UNO Rev 3 [2]

Table 2. Specifications of Arduino UNO Rev 3 [2]

Microcontroller	ATmega328P
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
PWM Digital I/O Pins	6
Analog Input Pins	6
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328P) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328P)
EEPROM	1 KB (ATmega328P)
Clock Speed	16 MHz
LED_BUILTIN	13
Length	68.6 mm
Width	53.4 mm
Weight	25

D. *TM1637 LED display*

The TM1637 LED display is a 7 segment 4 digit LED screen integrated with the TM1637 chip which functions to control the LED display. This screen has 4 pins [3]:

- VCC
- GND
- CLK - Clock; connected to the digital pin of the Arduino
- DIO - Data I / O; connected to the digital pin of the Arduino

Figure 4. LED TM1637 Display [3]

E. *Raspberry Pi 3 Model B*

The Raspberry Pi is a single-board computer made by the Raspberry Pi Foundation that is used to teach basic computer science in schools and in developing countries. [4]

The Raspberry Pi used for this study was the Raspberry Pi 3 Model B.

Figure 5. Raspberry Pi 3 Model B [4]

Table 3. Specifications of the Raspberry Pi 3 Model B [4]

Processor	Broadcom BCM2837 64bit ARMv7 Quad Core Processor powered Single Board Computer running at 1.2GHz
RAM	1 GB
Fitur	• BCM43143 WiFi on board • Bluetooth Low Energy (BLE) on board • CSI camera port for connecting the Raspberry Pi camera • DSI display port for connecting the Raspberry Pi touch screen display • Upgraded switched Micro USB power source (now supports up to 2.4 Amps) • Expected to have the same form factor has the Pi 2 Model B, however the LEDs will change position • Full size HDMI
Pin	40pin extended GPI
USB	4 x USB 2 ports
Audio	4 pole Stereo output and Composite video port
Storage	Micro SD port for loading your operating system and storing data

F. Design of Production Machine Output Monitoring Tool

The design of production machine output monitoring equipment is divided into two parts; a tool for reading engine output and software for monitoring on the Raspberry Pi 3 Model B. The following is a picture that shows a production machine output flowchart.

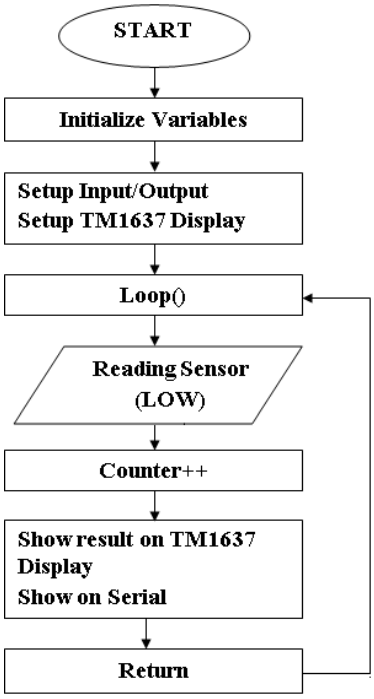


Figure 6. Flow Chart of Engine Output Reader

Next, here is a picture that shows a production machine output flowchart.

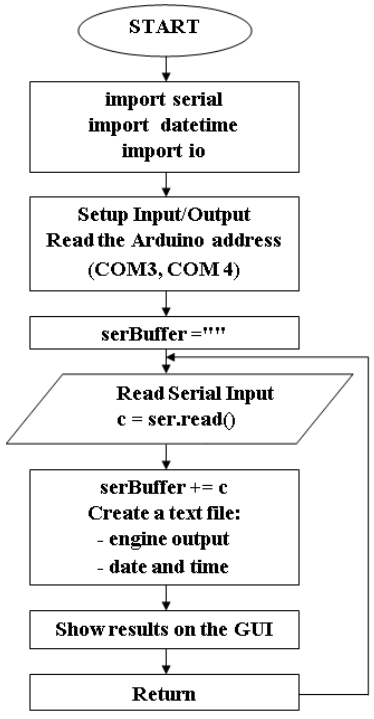


Figure 7. Flow Chart of the Machine Output Monitoring Program

III. RESULTS AND DISCUSSION

A. Engine Output Reader

This production machine output reader will be placed in the output section of the production machine to calculate the amount of output from the machine. The Ardunino pins used are pins 8, 9, and 10. Pins 8 and 9 are connected to the DIO and CLK of the TM1637 LED Display. Pin 10 is used to

receive digital input from the sensor E18 D80NK. Each Vcc TM1637 LED Display and Sensor E18 D80NK are connected to a 5V voltage source that can be taken from the Arduino. The GND TM1637 LED Display and the Sensor E18 D80NK are also connected to the GND on the Arduino.

The image below shows the schematic of the engine output reader.

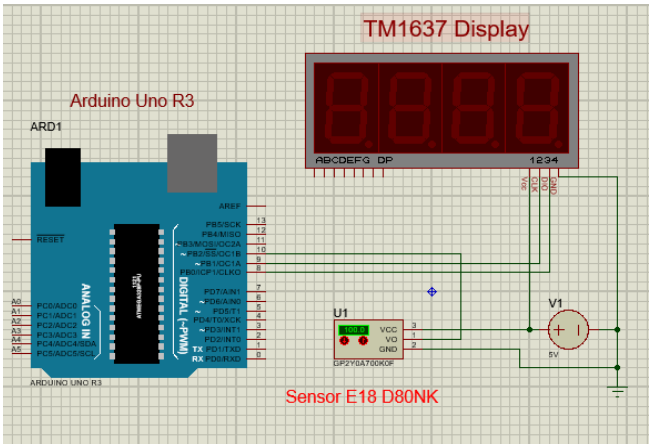


Figure 8. Schematic of Engine Output Reader

After the tools are assembled, a program for Arduino is created using the Arduino software (IDE). This program functions to calculate the number of outputs from the production machine. The Sensor E18 D80NK is active LOW so that when the sensor detects an object, the sensor will provide LOW input. When the Sensor E18 D80NK provides LOW input via pin 10, the program will add a counter value of 1. The result of adding this value will be displayed on the TM1637 LED screen and also sent to the Raspberry Pi via a USB connection. This program will be LOOP so that the counter will continue to count until the appliance is reset.

The following programs are used:

```
#include <TM1637Display.h>;
const int SENSOR=10;
const int CLK = 9;
const int DIO = 8;
int sensorValue = digitalRead(SENSOR);
int counter;
TM1637Display display(CLK, DIO);

void setup() {
  Serial.begin(115200);
  pinMode(SENSOR, INPUT_PULLUP);
 display.setBrightness(0x0a);
}

void loop() {

   while (
digitalRead(SENSOR)==LOW){DealWithSwitchPre
ss();};
};

void DealWithSwitchPress()
  {
delay(1);
    counter++;
 display.showNumberDec(counter);
 Serial.println(counter);
 while (digitalRead(SENSOR)==LOW){;};

delay(1);
```

B. Monitoring on the Raspberry Pi

There are many production machines used at PT. Tiga Serangkai Inti Corpora. These machines will be monitored in real time via a central computer. This central computer is the Raspberry Pi. To monitor these production machines, a program that can be run on the operating system of the Raspberry Pi is needed. The Python programming language is suitable for use in making this monitoring program.

PySerial needs to be added to Python 2.7 to read addresses and communicate with Arduino. The address of the Arduino can be seen from the Arduino Software (IDE). After knowing the address of Arduino, this address needs to be included in the program to be created. The program has a Graphical User Interface (GUI), which will read the input from two Arduinos. This is done to show that this Monitoring can be applied to a larger scale.

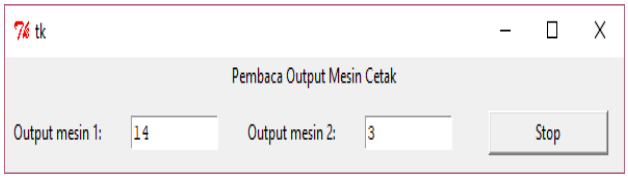
The Python program will read data from two Arduino series connected to the Raspberry Pi. When an input is entered, it will be displayed in the GUI and also saved into a text file. This text file contains the engine output data and the time when that output occurred. This program will be LOOP until the program is closed.

The Python program that is made is as follows:

```
import Tkinter as tk
from serial import *
from datetime import datetime
import io
 #ensure non-blocking
ser = Serial('COM3', 115200, timeout=0)
ser2 = Serial('COM4', 115200,
timeout=0)

root = tk.Tk()
serBuffer = ""
serBuffer2 = ""

def readSerial():
   while True:
      c = ser.read() # attempt to read a
character from Serial

      #was anything read?
      if len(c) == 0:
        break

      # get the buffer from outside of this
function
      global serBuffer

      if c == '\n':
         serBuffer += "\n"  # add the
newline to the buffer
```

```
            #add the line to the TOP of the
log
            log.insert('0.0', serBuffer)
            serBuffer = "" # empty the buffer

        else:
            serBuffer += c # add to the buffer
            outfile='Mesin1.txt'
            with open(outfile,'a') as f:

f.write(datetime.now().isoformat() + '\t' +
serBuffer + '\n')
                f.flush()
        root.after(1, readSerial) # check serial
again soon

    def readSerial2():
        while True:
            d = ser2.read() # attempt to read a
character from Serial

            #was anything read?
            if len(d) == 0:
                break

            # get the buffer from outside of this
function
            global serBuffer2

            # check if character is a delimeter
            if d == '\r':
                d = '' # don't want returns. chuck
it

            if d == '\n':
                serBuffer2 += "\n"  # add the
newline to the buffer

                #add the line to the TOP of the
log
                log2.insert('0.0', serBuffer2)
                serBuffer2 = "" # empty the buffer

            else:
                serBuffer2 += d# add to the
buffer
                outfile='Mesin2.txt'
                with open(outfile,'a') as f:

f.write(datetime.now().isoformat() + '\t' +
serBuffer2 + '\n')
                f.flush()

        root.after(1, readSerial2) # check
serial again soon

    root.after(1, readSerial)
    root.after(1, readSerial2)

    w = tk.Label(root, text="Pembaca
Output Mesin Cetak").pack()
    w2 = tk.Label(root, text="Output mesin
1:").pack(padx=5, pady=10,side=tk.LEFT)
    log = tk.Text ( root, width=10, height=1,
takefocus=0)
    log.pack(padx=20,
pady=10,side=tk.LEFT)

    w3 = tk.Label(root, text="Output mesin
2:").pack(padx=5, pady=10,side=tk.LEFT)

    log2 = tk.Text ( root, width=10,
height=1, takefocus=0)
    log2.pack(padx=20,
pady=10,side=tk.LEFT)

    tombol = tk.Button(root, text='Stop',
width=15, command=root.destroy)
    tombol.pack(padx=15, pady=10,
side=tk.LEFT)

root.mainloop()
```

Furthermore, the image below is the GUI display of the production machine monitoring program that will be operated on the Raspberry Pi.



And the results of monitoring data storage from the engine output are stored in the form of a txt file, as shown in the following figure:

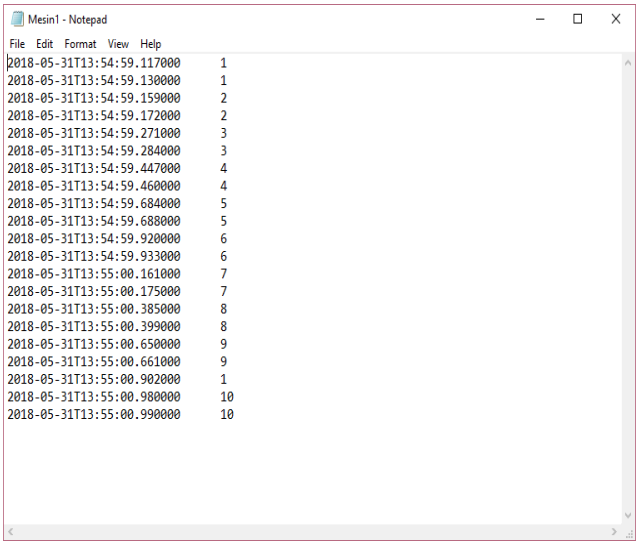


Figure 10. The results of the production machine output monitoring data storage

IV. CONCLUSION

The production engine output monitoring tool consists of a central computer which is the Raspberry Pi 3 model B and a program in the Python programming language. This program will display the input from Arduino UNO R3 which is communicated via USB on the GUI and also saves this

input data into a text file. Arduino devices must be defined and addressed manually in the coding of the Python program. This GUI program is still in hard code form, this program does not have its own database, there is no plug and play feature for machine output readers. In storing the data into a text file, there is a repetition of the stored data which is most likely the result of two LOOPING programs. The first LOOP occurs in the Arduino and the second LOOP occurs in a Python program.

ACKNOWLEDGMENT

The authors are grateful to the IT Infrastructure Division of PT. Tiga Serangkai Inti Corpora, Surakarta, Indonesia for conducting this research.

REFERENCES

[1] _____, Proximity Sensor/Switch E18-D80NK, https://www.rhydolabz.com/documents/27/E18-D80NK.pdf Accessed 17 Januari 2021

[2] _____, Arduino UNO Rev 3, https://store.arduino.cc/usa/arduino-uno-rev3 Accessed 17 Januari 2021

[3] _____, TM1637, https://playground.arduino.cc/Main/TM1637/ Accessed 17 Januari 2021

[4] _____, Raspberry Pi 3 Model B, https://www.raspberrypi.org/products/raspberry-pi-3-model-b/ Accessed 17 Januari 2021