The main controller of the printer is a Raspberry Pi 4B (RPi). The RPi will be in constant communication with the rover on which it is mounted and is responsible for controlling the printing process.

Startup

Logging

Upon startup the RPi will switch on and the printer program will automatically run. A log file will be created, and the program start time will be logged. Throughout operation of the printer several pieces of information will be logged to this file. This log file can be used to track any errors that the printer encounters.

Communication

The RPi will have two channels of communication. The first will be with the rover’s main computer through TCP/IP over ethernet and the second will be with the Arduino over USB communication.

Once the program has started the RPi will initiate TCP/IP communication as the server and wait for the rover to connect. The time that the RPi initiates the communication is logged.

All communication between the RPi and the rover will be done using XML as shown in Appendix X.

Handshake

When the rover connects it is logged and the RPi will send a handshake message (Appendix X) and wait for a response. The handshake response from the rover (Appendix X) will tell the RPi if the printer will be functional on each occasion, as the rover will not always need to use the printer. In the case that the printer will not be functional, then the RPi will do nothing. If the printer is to be functional, then the RPi will send a message to the Arduino to prime the printer with ink. Once the priming is complete, the Arduino will send a “Ready” message to the RPi which will then send a “Ready” message to the rover. These events will each be logged.

Heartbeat (Status update)

Once communication has been established, the rover and RPi will send status updates (Appendix X) to each other at a frequency of 30 Hz. These status updates (or heartbeats) will contain relevant information needed to ensure smooth operation. The rover will send the current speed of the rover, the distance moved since the last heartbeat, any errors that occurred, and whether the job is still active. The RPi will send any errors, whether the job is active, and if the printer is busy or available.

As heartbeats come in from the rover, the RPi will update any relevant global variables so that other threads are using the current status.

Commands

As the rover moves around the site to different locations to print it will send print commands (Appendix X) to the printer at these locations. Each print command will detail what is to be printed next. The information contained in a command includes the following:

1. the command ID
2. length of the line or lines
3. text to be printed
4. direction the rover will move
5. the type of line to be printed
6. a parameter detailing which position to start from (this is in case of an interruption in a previous attempt to print a command).

This information is assigned to global variables which are used in the command processing stage.

Command Processing

When command variables are updated with a new command they are translated into a pattern of ones and zeros for the piezoelectric pumps to be actuated. This is achieved by creating an array of decimal values that will be shifted into the 74HC595 shift register connected to the piezoelectric pump circuitry. The array is created in the following steps:

1. The code loops through the text to be printed and for each character in the string it appends the pattern associated to that character in the characters dictionary (Appendix Xb) to a new array.
2. The number of elements of the array is increased or decreased to match the length of the line as specified in the command. The distance between each column is also determined at this stage to ensure that the final dot is printed at the correct distance.
3. The line type detailed in the command is masked over the text.
4. The bits of the elements in the array are offset at a 45 angle to match the angle of the printhead according to the direction that the rover is moving.

The array and the distance between each element are passed on to the printing stage next.

Printing

Once the pattern array has been created the rover is alerted that the printer is ready to print and the RPi waits for the rover to send a print trigger (Appendix X). A separate thread will wait for the trigger, once the trigger occurs the array is passed to the printing stage of the program, which takes one element at a time and shifts it into the 74HC595 shift register. The output enable pin (Active-LOW) of the shift register is set low for 2.5 ms and then set high again. The delay between one element and the next is determined by the current speed of the rover (from heartbeat) and the set distance between the elements.

As each element is printed it is logged. In the event of an interruption during the print the RPi can calculate which element to continue printing from, the rover can also specify which element to start printing from.

Shutdown

Once the printing is complete the rover will send an update to tell the RPi to shutdown. The RPi will send a “PURGE” message to the Arduino which will purge and then send a “PURGE COMPLETE” message to the RPi. The RPi will send a “PURGE COMPLETE” message to the rover followed by the log file. The RPi will then shut itself down.

Appendix X

<?xml version='1.0' encoding='UTF-8'?>

<Blueprint>

<Handshake>

<Functional>**YES**</Functional>

<Functional>**NO**</Functional>

</Handshake>

</Blueprint>

<?xml version='1.0' encoding='UTF-8'?>

<Rugged>

<Handshake>

<Functional>**YES**</Functional>

</Handshake>

</Rugged>

<?xml version='1.0' encoding='UTF-8'?>

<Blueprint>

<Handshake>

<Functional>**READY**</Functional>

</Handshake>

</Blueprint>

<?xml version='1.0' encoding='UTF-8'?>

<Rugged>

<Status>

<Activity>**Active**</Activity>

<Errors>**0**</Errors>

<RoverSpeed>**0.5**</RoverSpeed>

<Distance>**1**</Distance>

</Status>

</Rugged>

<?xml version='1.0' encoding='UTF-8'?>

<Blueprint>

<Status>

<Activity>**Active**</Activity>

<Errors>**0**</Errors>

<Availability>**Available**</Availability>

</Status>

</Blueprint>

Example command from Rover

<?xml version='1.0' encoding='UTF-8'?>

<Rugged>

<Command>

<id>**1**</id>

<Text>**TEXT GOES HERE**</Text>

<Lines>**4**</Lines>

<Direction>**N**</Direction>

<Distance>**1500**</Distance>

<Begin>**1**</Begin>

</Command>

</Rugged>

<?xml version='1.0' encoding='UTF-8'?>

<Blueprint>

<Command>

<Activity>**Processing**</Activity>

</Command>

</Blueprint>

<?xml version='1.0' encoding='UTF-8'?>

<Rugged>

<Trigger>

<Start>**GO**</Start>

</Trigger>

</Rugged>