ArrayBot

Smith Lab, Allen Institute - 2016 - 2019

Part One

1. Overview of ArrayBot Software Applications

1.1 Introduction

This document gives an overview of the software that is named *ArrayBot*.

The ArrayBot software is designed to control a set of motorized stages, a camera and other hardware, in order to assist in the process of collecting biological tissue ribbons produced by an ultra-microtome.

The ArrayBot software is partitioned into a set of specialized software applications, ArrayBot, ArrayCam and the ArduinoController application.

The ArrayBot application is designed to interface with the motorized part of the ArrayBot hardware. The hardware can be divided into two roughly identical XYZ stages, the *coverslip* stage and the *whisker stage* (picture).

the ArrayBot UI do expose control interfaces to all motor hardware as well as a user interface for programming automated motor sequences. Automation is discussed in a later section.

In addition, the UI do also contain elements for ribbon separation, and ribbon length control.

The ArrayCam application expose and interface with the ArrayBot camera, as well as certain control of peripheral lights related to the camera vision. The ArrayCam application allow the user to take camera snapshots as well as recording movies.

The Arduino controller application purpose is to communicate with a set of Arduino boards, and forward the controls of these to the ArrayBot and ArrayCam applications.

The ArduinoController is designed to act as a server of its connected (Arduino) hardware.

The following section discusses these applications in greater detail.

1.2 Arraybot UI's

1.2.1 ArrayBot UI

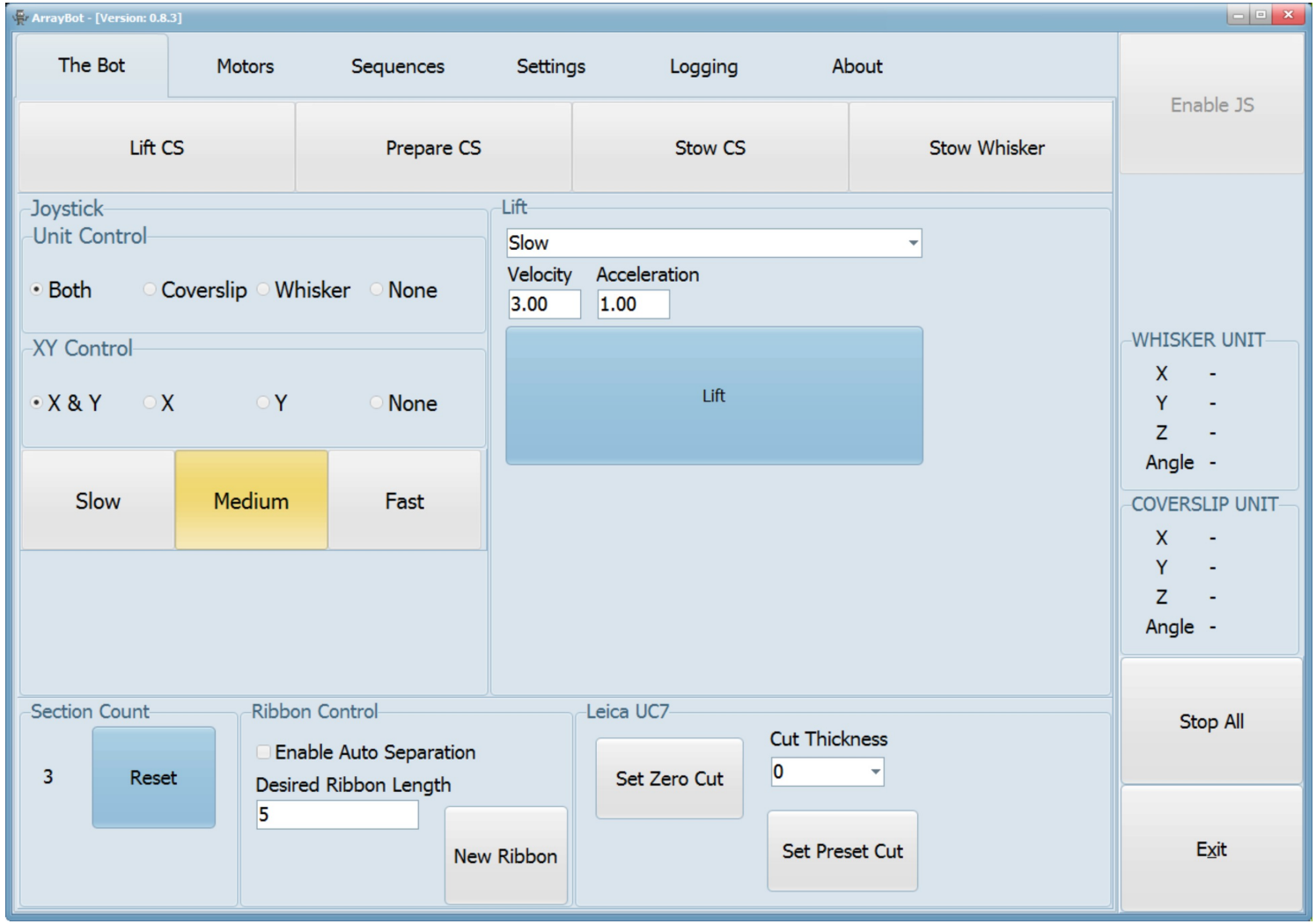

Figure 1.1: ArrayBot UI

1.2.2 ArrayCam UI

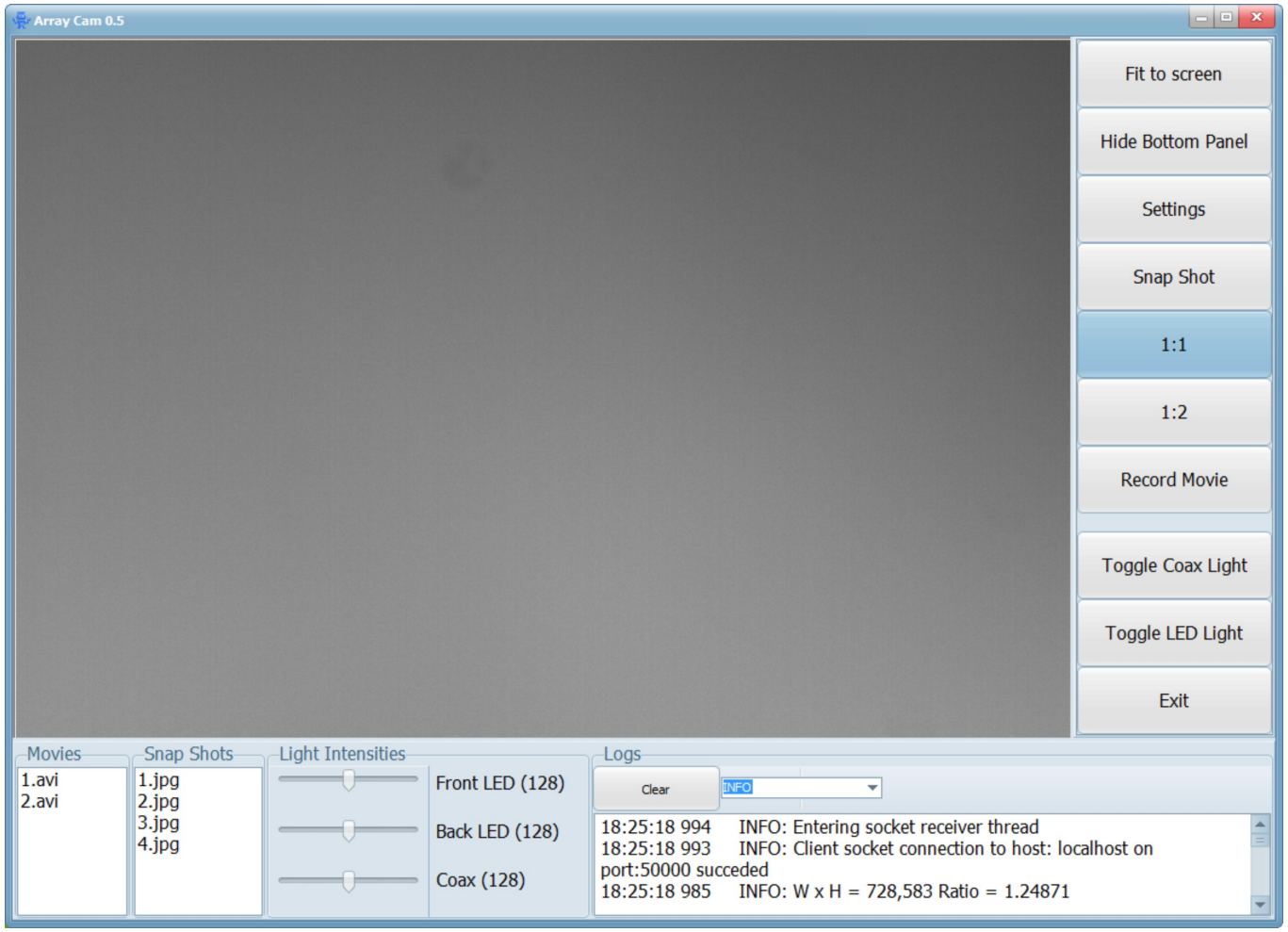

Figure 1.2: ArrayCam UI

1.2.3 ArduinoController UI

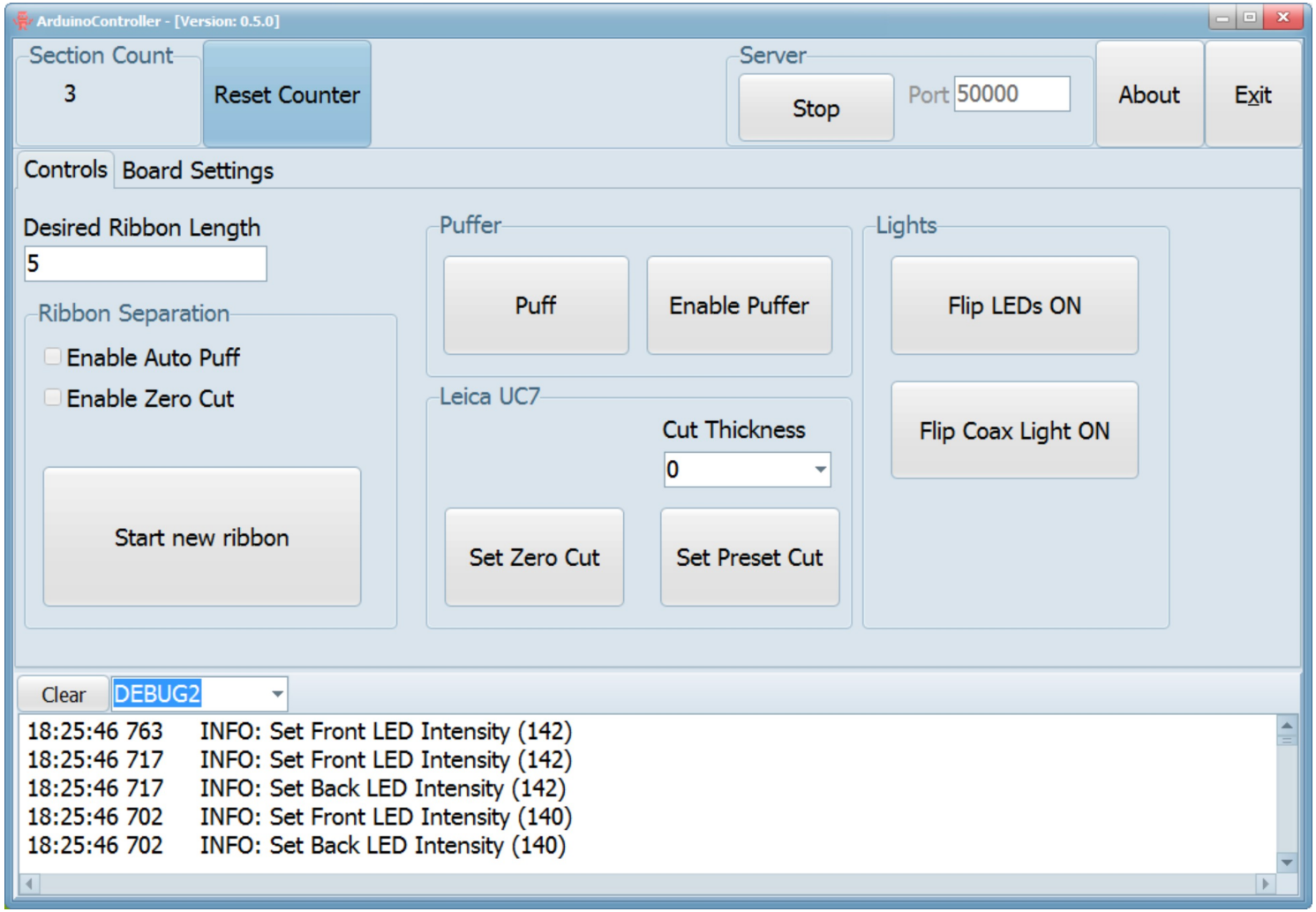

Figure 1.3: ArduinoController UI

1.3 Arraybot Hardware Setup

1.3.1 Motors

1.3.2 Arduinos

1.3.3 The Camera

2. Software Design and Software Components

2.1 Rough Software setup - picture

3. IPC

4. Process Sequencer

The process sequencer component is designed to allow the user to define and execute a sequence of various actions, e.g. moves.

4.0.1 Process Sequence

4.0.2 Process

4.0.3 XML

4.0.4 Triggers

5. The JoyStick

5.0.1 JoyStickMessageDispatcher

6. Controlling the motors

7. The Arduino Server

The Arduino server class encapsulate Arduino peripherals in the Arraybot system, i.e. the sensor, puffer and the Leica Arduinos. Clients are served with Arduino related messages over a TCP/IP socket.

The Arduino server is a descendant of the IPC server class that is implementing all network functionality.

The Arduino server forwards any messages sent from the arduino board to connected TCP/IP clients.

The purpose of each one of the three Arduinos, the Puffer board, the Sensor board and the Leica Arduino board, are discussed in sections below.

7.1 Messages exchanged with the Arduino Server

This section discusses the messaging protocol for the Arduino Server. The protocol is defined by a few simple text messages (short strings) that can be exchanged with the server.

Typically a client sends (1) a command to the server, e.g. "ENABLE_PUFFER", (2) the server receives and handles the message, possibly causing peripheral hardware to be modified, and (3) a response may be sent back to the client.

The design emphasizes simplicity and so the error checking implemented in the system is held

to a minimum. For example, a client sending a request do not typically implement a mechanism to handle a server response notifying if the request was carried out or not. Instead, this system rely on careful debugging by the user, in reading error and warning log messages.

7.1.1 Incoming messages

Table 7.1 lists commands that the Arduino server accepts.

Text Message	Args	Description
RESET_SECTION_COUNT	none	This message is a request to reset the current section count.
ENABLE_PUFFER	none	Request to enable the puffer.
DISABLE_PUFFER	none	Request to disable the puffer.
ENABLE_AUTO_PUFF	none	Enable AutoPuff mechanims.
DISABLE_AUTO_PUFF	none	Disable AutoPuff mechanims.
SET_DESIRED_RIBBON_LENGTH	none	Set desired ribbon length. Used when the AutoPuff mechanism is enabled.
PUFF	none	Request a (manual) puff.
START_NEW_RIBBON	none	Request initiation of a new ribbon. This command will restore the current cut thickness preset and reset the section count.
TURN_ON_LED_LIGHTS	-	Turn on LED's
TURN_OFF_LED_LIGHTS	-	Turn off LED's
TURN_ON_COAX_LIGHTS	-	Turn on Coax lights
TURN_OFF_COAX_LIGHTS	-	Turn on Coax lights
TOGGLE_LED_LIGHT	none	Toggle LED light on/off.
TOGGLE_COAX_LIGHT	none	Toggle Coax light on/off.
SET_FRONTLED	none	Set Front LED intensity.
SET_BACKLED	none	Set Back LED intensity.
SET_COAX	none	Set Coax LED intensity.
SET_CUT_PRESET	none	Set cut thickness preset.
SET_DELTAY	none	Set mouse delta Y on the Leica UI.
GET_SENSOR_ARDUINO_STATUS	none	Get general server status.
SENSOR_CUSTOM_MESSAGE	none	
GET_PUFFER_ARDUINO_STATUS	none	
GET_SERVER_STATUS	none	

Table 7.1: Incoming messages handled by the Arduino server

7.1.2 Outgoing messages

Text Message	Initiated by	Description
GET_READY_FOR_ZERO_CUT_1	Puffer Arduino	This message is sent two counts before a zero cut is to be executed.
GET_READY_FOR_ZERO_CUT_2	Puffer Arduino	This message is sent one counts before a zero cut is to be executed.
SET_ZERO_CUT_1	Puffer Arduino	This message is sent when a zero cut is to be executed.
RESTORE_FROM_ZERO_CUT	Puffer Arduino	This message is sent after a zero cut.

Table 7.2: Arduino server response messages

7.2 The Puffer and ribbon separation

Individually cut sections are held together by a thin film of glue. The glue is part of the 'block-phase' and is necessary in order to successfully create a ribbon of consecutive tissue sections. However, the glue may also make a cut section stick to the knife itself.
In order to separate a ribbon from the knife, as to give way to a new ribbon, a mechanism involving an air puffer together with a zero thickness cut, allows for automatic separation of a ribbon from the knife.

The puffer mechanics is controlled by an Arduino, and the zero cut is controlled by serial communication between the Leica ultra-microtome and the host computer.

7.2.1 Puffer Arduino Communication Protocol

The Puffer Arduino hardware are connected to the ArrayBot software using a serial (COM) port. This allow messaging between the PC software and the Arduino Software.

The Puffer Arduino implements a simple messaging protocol, involving receive/transmit of a few bytes over the serial port connection.

Table 7.3 lists the commands that are currently implemented in the Puffer Arduino Software.

Command	Arguments	Usage	Note
S	1 or 0	Enable/Disable simulation of the Hall Sensor.	For debugging purposes.
s	int	Set simulation speed. The supplied integer denotes the delay, in ms, between sending simulated Hall sensor switch messages.	For debugging purposes.
P	int	Set cut thickness preset on the Leica.	This command results in a request to the Leica Arduino to change the cut thickness preset. Valid values are 1-5
Y	int	Set deltaY for mouse movement on the Leica.	
e		Enable the puffer	
a		Disable the puffer	
p		Request an immediate puff	
d	int	Set puff duration in ms	
v	int	Set puff valve speed. Valid values are 0-255.	
i	int	Request info about current Arduino state.	

Table 7.3: Puffer Arduino commands

Message	Usage	Note
LEICA MESSAGE: "..."	Forwarded message from the Leica Arduino	For debugging.
HALL_SENSOR='VALUE'	Message indicating the value of the Hall Sensor. VAL = HIGH or LOW	
REQUEST_CUT_PRESET_'VALUE'	Indicating we are requesting to change the cut thickness preset on the Leica. VAL = 1-5	
CUT_PRESET_VALUE IS INVALID	Error response	
REQUEST_DELTA_Y='VALUE'	Indicating request to change the delta Y value on the Leica. VAL = any integer	
PUFFER_ENABLED	Indicate that the puffer was enabled	
PUFFER_DISABLED	Indicate that the puffer was disabled	
EXECUTED_PUFF	Puffer was executed	

Table 7.4: Puffer Arduino Client Messages

7.3 Sensors and Lights

Lights and sensors are read/controlled by a third Arduino - the 'Sensor' Arduino.

Table 7.6 lists the commands that are currently implemented in the Sensor Arduino Software.

Command	Arguments	Usage	Note
1	-	Button1 Down	Turn on light?
2	-	Down	
3	-	Down	
3	-	Down	
4	-	Down	
5	-	Down	
c	int	Set coax drive 0-255	
f	int	Set front LED drive 0-255	
b	int	Set back LED drive 0-255	
i		Return information about states	

Table 7.5: Sensor Arduino Commands

Response	Description
BUTTON_1_DOWN	Turn off Coax LEDS
BUTTON_2_DOWN	
BUTTON_3_DOWN	
BUTTON_4_DOWN	
BUTTON_5_DOWN	
COAX_DRIVE=	
FRONT_LED_DRIVE=	
BACK_LED_DRIVE=	
PIN_1_DOWN	
DHT22_DATA=	
DHT22_ERROR	

Table 7.6: Sensor Arduino Responses

7.4 Hardware Setup

7.4.1 Arduino setup

The figure and tables below discusses the hardware setup of the three Arduinos, the PC and the leica microtome

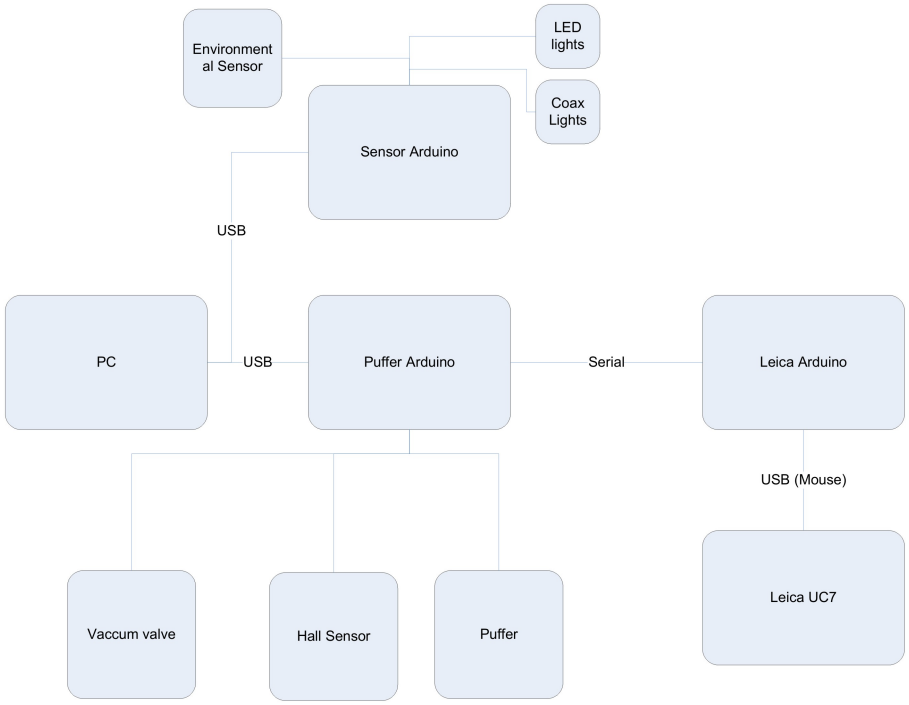

Figure 7.1: Arduino hardware setup

Appendices

A. Get the source code

Public Software Repository: **git@github.com** : **TotteKarlsson/ArrayBot.git**

B. Software API's

C. Arduino wiring

Pin #	Usage	Note
0	Serial Receive	Used for serial communication
1	Serial Transmit	Used for serial communication
2	Hall Sensor	
3	Puffer	
4		
5		
6		
7		
8		
9		
10	Serial Receive	Connection to the Leica Arduino Serial port
11	Serial Transmit	Connection to the Leica Arduino Serial port
12		
13	LED Illuminates when Hall sensor is high	

Table C.1: Puffer Arduino Pins

Pin #	Usage	Note
0	Serial Receive	Used for serial communication
1	Serial Transmit	Used for serial communication

Table C.2: Sensor Arduino Pins

Pin #	Usage	Note
0	Serial Receive	Used for serial communication to Puffer Arduino
1	Serial Transmit	Used for serial communication to Puffer Arduino
2	unused	
3	unused	
4	unused	
5	unused	
6	unused	
7	unused	
8	unused	
9	unused	
10	unused	
11	unused	
12	unused	
13	unused	

Table C.3: Leica Arduino Pins