ECE375 Lab 1

TA: Dongjun Lee

School of Electrical Engineering and Computer Science

Oregon State University

Lab Information

- Canvas
  - All lab materials can be found.
    - Lab Handouts and PPT slides for the lab instructions.
    - Skeleton Codes (and Example Codes, if available)
    - Simulator Installation Guides
    - Datasheets and AVR Instruction Set Manual
  - Assignments should be submitted via Canvas.
  - Gradings for each lab will be updated by the beginning of the following week's lab.

Must-know Policies

- COVID-19
  - Face covering is recommended.
  - Stay home and alert your TA when you test positive or have symptoms.

- Lab sessions
  - Will be used for check-offs.
  - Only your lab TA can get you checked off.
  - Check-offs are based upon the principle of students, who must be present on site.

- Office Hours
  - TAs will run their office hours at the lab.
  - Asking for debug-helps must be gone through office hours or separate appointments.
    - Note, we do not respond to your email that contains code.
    - Because we need to check both hardware and software to examine.

Work Policy

- You can work alone or in a group as you prefer.
  - A group can have no more than two people.
  - Your partner should belong to the same lab session.
  - You can change the partner for each lab.

- Submission
  - Every individual must submit assignments via Canvas.
  - Must include your (and partner's) names to prevent plagiarism issues in
    - Comments in the beginning of the **CODE**.
    - File name of the **CODE**.
  - You and your partner must submit the exact same code file.

- No late work is allowed.
  - 2 weeks labs: Lab 1 and Lab 7.
  - 1 week labs: Lab 2 – 6.

- Recycling your work from the previous terms is **forbidden**. Otherwise, it will flag plagiarism issue.

Plagiarism Policies

- Our policy is directly reporting you to the department as academic dishonesty.

- If you're in a group, you and the partner is on joint responsibility.
  - It's also your obligation to manage your partner.

- Plagiarism checking will happen at a random time.
  - Don't waste your whole work of a term by falling into one-time temptation.

- It is never allowed to represent another person's work as your own even just the small part.

Plagiarism includes

- Excluding your partner’s name.
    - The plagiarism checker will raise a red flag when you both submit the same file but does not specify both names.
- Recycling your code from the previous terms.
    - You can’t reuse the previous work of yours or your partner’s.
- Partially picking or copying code.
    - Cherry picking for code is a bad temptation.
    - We don’t care whether you understand the code or not.
- Deceiving by changing names of registers or comments.
    - Don’t presume the plagiarism checker is stupid enough.

Gradings

- Lab: 30 % of the course grade
    - Lab 1 – 6: 3.75% of the course grade
    - Lab 7: 7.50% of the course grade

Lab 1 Introduction

- Lab1 is to let you familiarized with software and the board. You are given 2 weeks.
  - Follow the **Installation Guides** in the Lab 1 Materials.

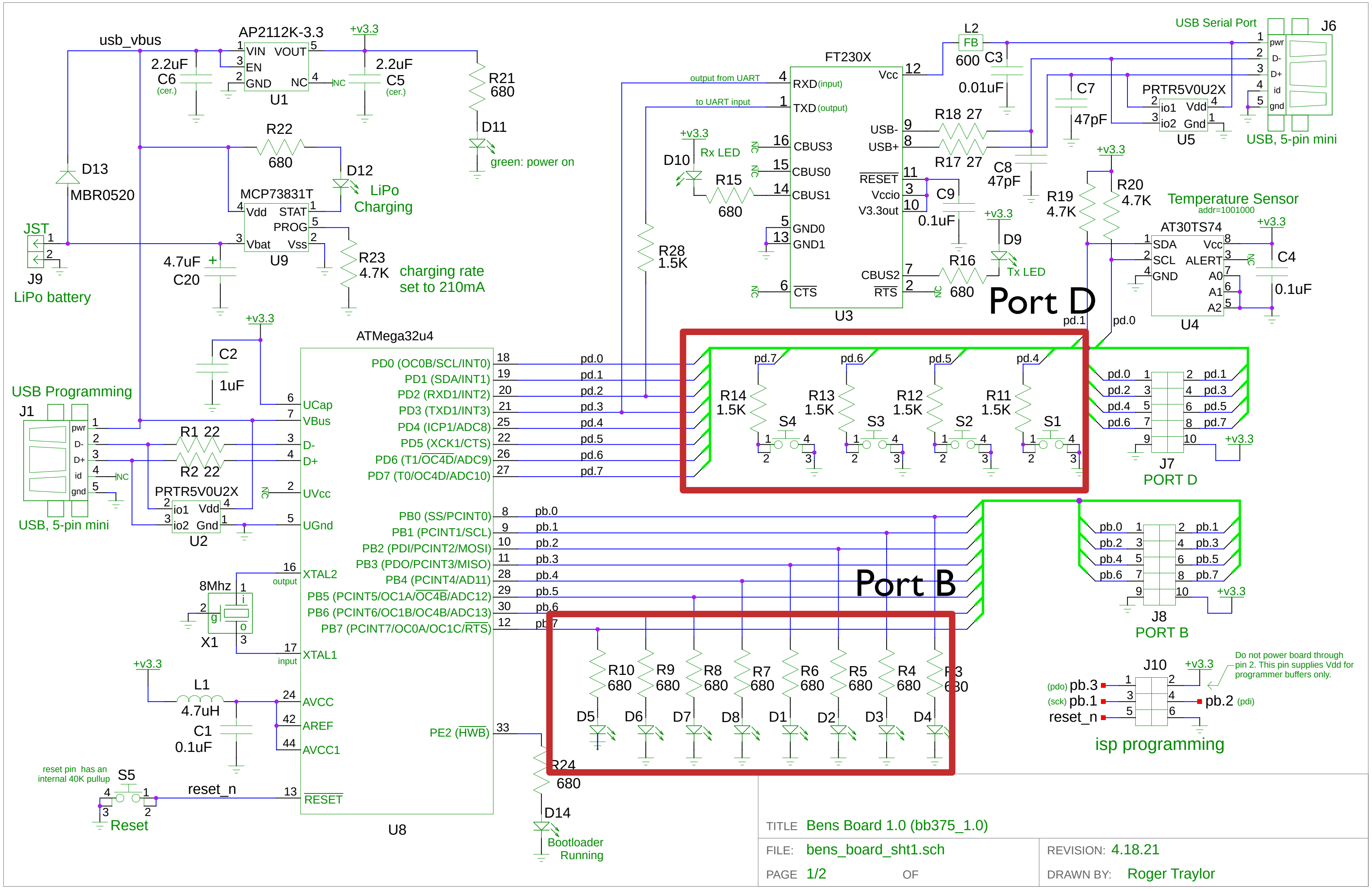
- **Part 1** (week 1)
  - Download the BasicBumpBot.asm code and program in your AVR board.
  - **Figure out** how TekBot **reverses for twice as long** before turning away and resuming forward motion. Demonstrate it during Demo session.

- **Part 2** (week 2)
  - Download the DanceBot.c code and understand how to configure the I/O ports.
  - Write a simple C program to replicate the bumptbot behavior in part 1. Demonstrate it during Demo session.

- Attend your checkoff session and demonstrate your work. Missing the checkoff session will result in a score of 0 for the lab grade.

Lab1 Check-off Instructions

- Download your submitted asm and C codes in Canvas in front of your Lab TA.
- Compile and flash the code to an AVR board.
- Demonstrate it’s correctly working.
  - Even if you didn’t make it to complete the lab, you still need to show your work to TA to get partial credits.
- Explain your code.
  - TA will ask some questions regarding the code. Poor answering will take away some credits.
  - Adding comments in every line is required.
- Answer Study Questions.
  - TA will ask some questions regarding the study questions in the lab handout. Poor answering will take away some credits.



TITLE Bens Board 1.0 (bb375_1.0)

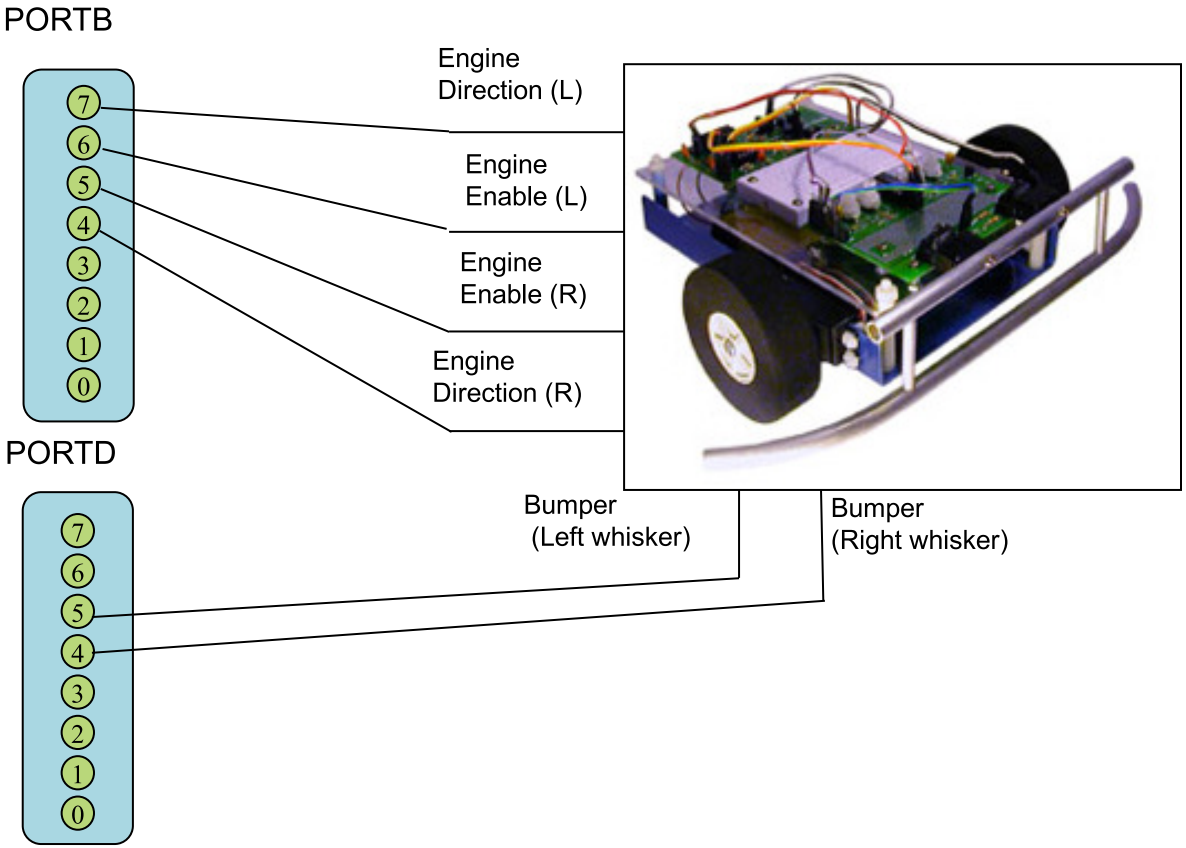
FILE: bens_board_sht1.sch

REVISION: 4.18.21

PAGE 1/2 OF

DRAWN BY: Roger Traylor

Connection Guides



PORTB

Engine Direction (L)

Engine Enable (L)

Engine Enable (R)

Engine Direction (R)

PORTD

Bumper (Left whisker)

Bumper (Right whisker)

Bumpbot Behaviors

● On(1) ○ Off(0)

- Forward

Left Motor Forward			Right Motor Forward				
● Bit 7	○ Bit 6	○ Bit 5	● Bit 4	○ Bit 3	○ Bit 2	○ Bit 1	○ Bit 0
	Left Motor Enable	Right Motor Enable					

- Halt

Left Motor Forward			Right Motor Forward				
●	●	●	●	○	○	○	○
	Left Motor Disable	Right Motor Disable					

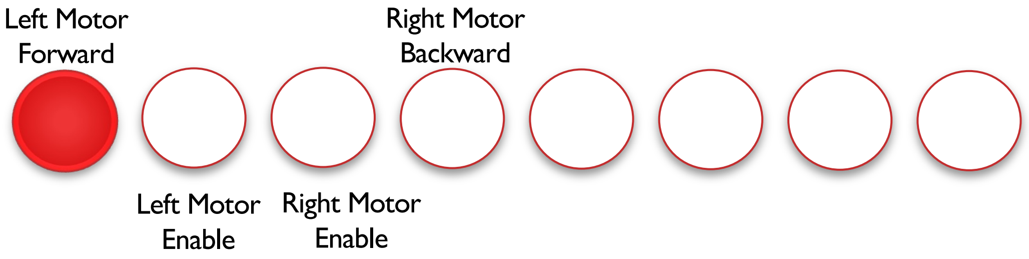
- Backward

Left Motor Backward			Right Motor Backward				
○	○	○	○	○	○	○	○
	Left Motor Enable	Right Motor Enable					

Bumpbot Behaviors

● On(1) ○ Off(0)

- Turn Right



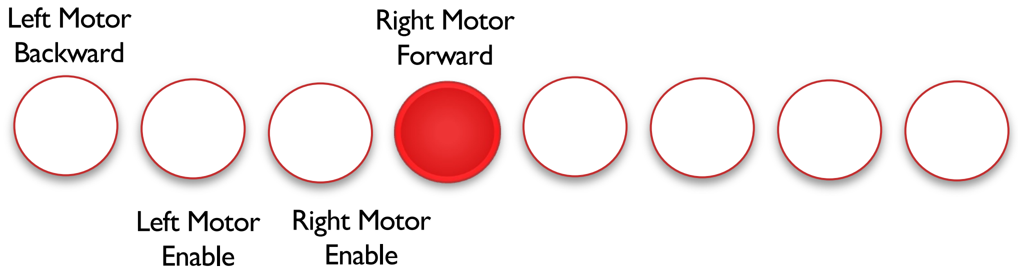
Left Motor Forward

Right Motor Backward

Left Motor Enable

Right Motor Enable

- Turn Left



Left Motor Backward

Right Motor Forward

Left Motor Enable

Right Motor Enable

Controlling Registers

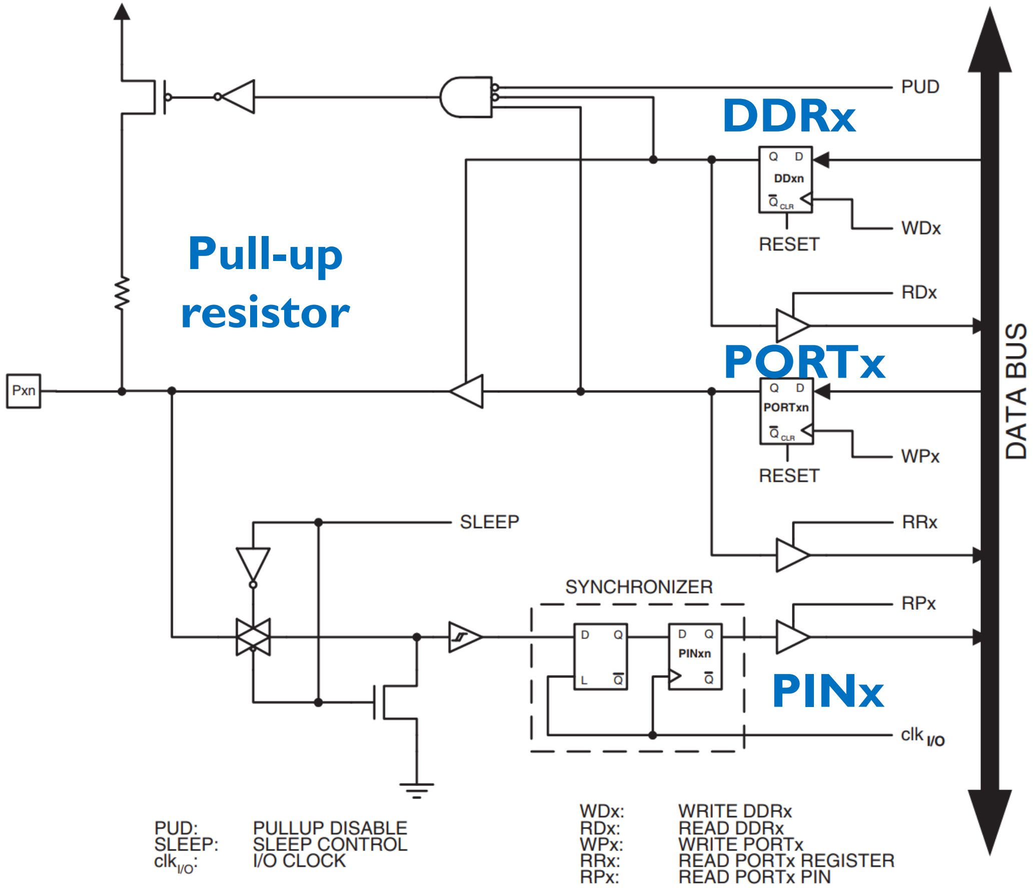
- Three types of registers
  - DDRx is a Data Direction Register for Port x
  - PORTx is a Port Output RegisTer for Port x
  - PINx is a Port Input register for Port x

- Output Port Settings
  - DDRB = 0b11111111 ; set 7-0 bits as outputs
  - PORTB = 0b11110000 ; turn on LEDs connected to 7-4 bits

- Input Port Settings
  - DDRD = 0b00000000 ; set 7-0 bits as inputs
  - PORTD = 0b11111111 ; enable pull up resistor
  - IN mpr, PIND ; read input data to mpr

AVR Ports

Figure 30. General Digital I/O[1]

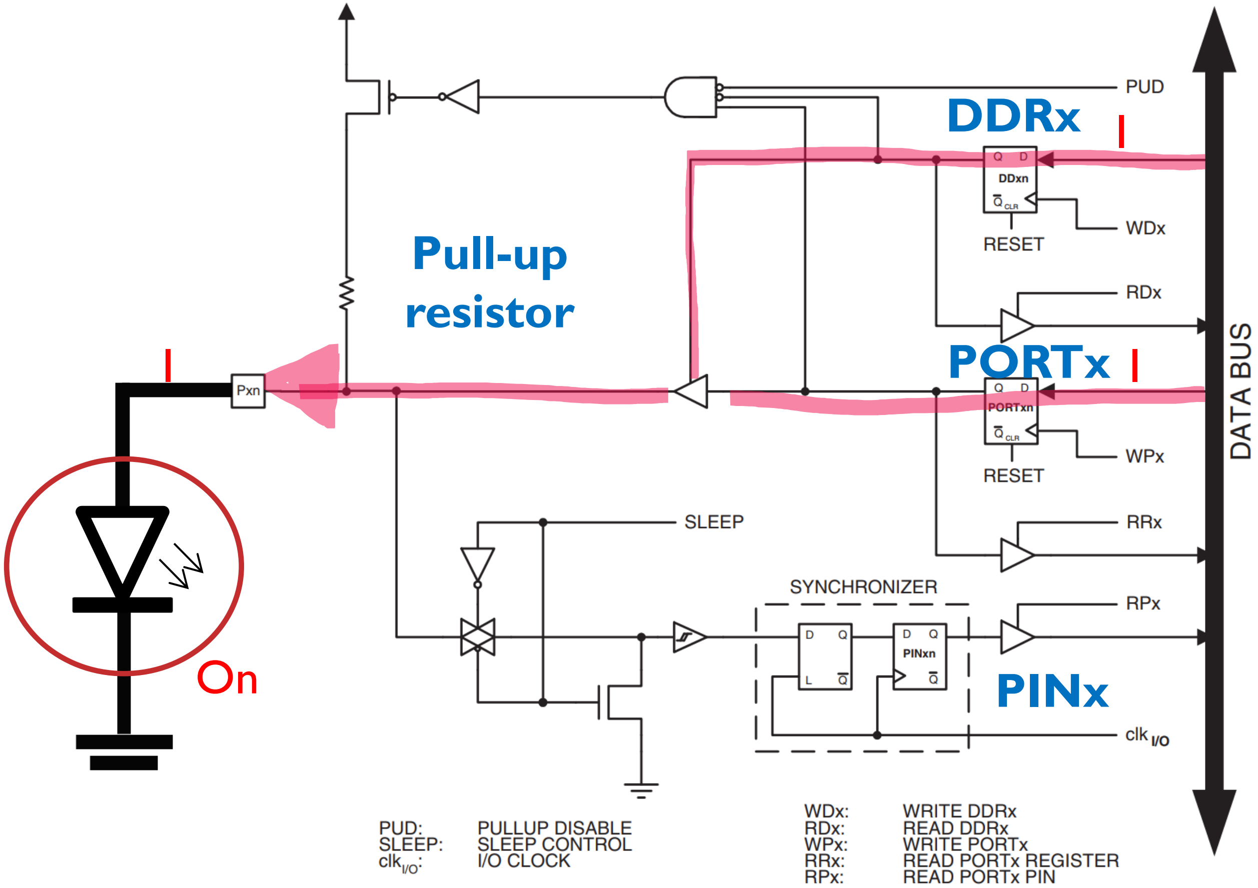


Note: 1. WPx, WDx, RRx, RPx, and RDx are common to all pins within the same port. $clk_{I/O}$, SLEEP, and PUD are common to all ports.

Atmega32U4 Datasheet 67p

AVR Ports – Configure output

Figure 30. General Digital I/O[1]



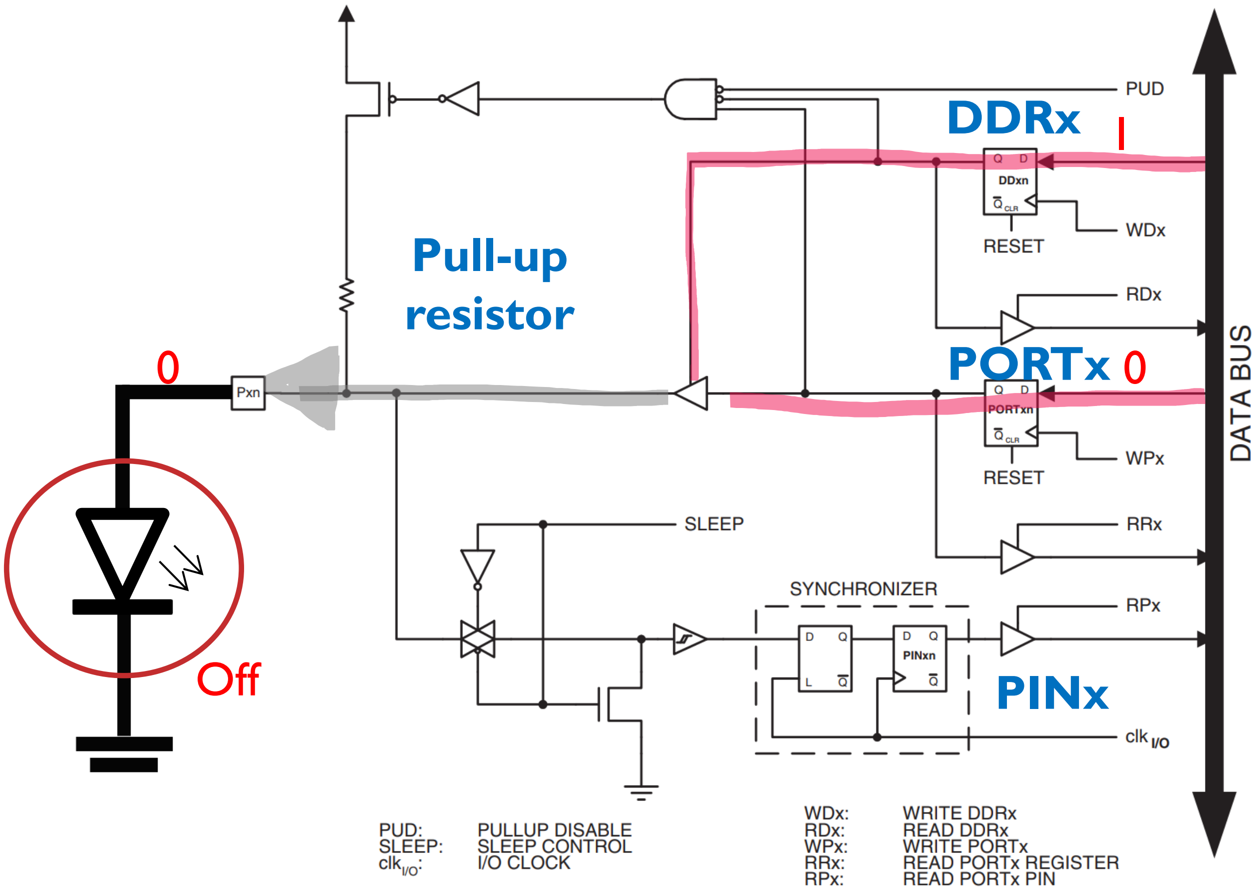
PUD: PULLUP DISABLE
SLEEP: SLEEP CONTROL
$clk_{I/O}$: I/O CLOCK

WDx: WRITE DDRx
RDx: READ DDRx
WPx: WRITE PORTx
RRx: READ PORTx REGISTER
RPx: READ PORTx PIN

Note: 1. WPx, WDx, RRx, RPx, and RDx are common to all pins within the same port. $clk_{I/O}$, SLEEP, and PUD are common to all ports.

AVR Ports – Configure output

Figure 30. General Digital I/O[1]

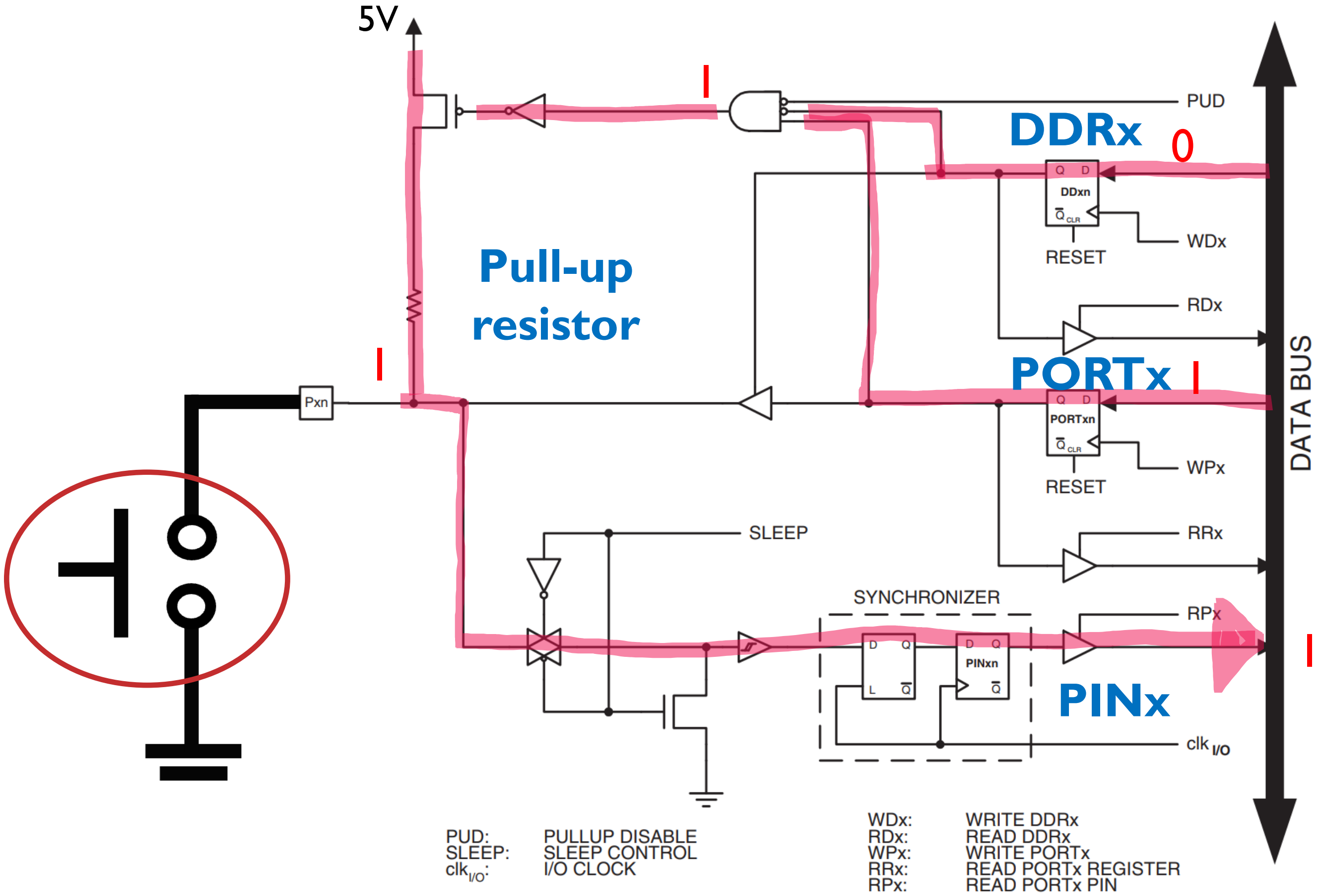


PUD:	PULLUP DISABLE
SLEEP:	SLEEP CONTROL
$clk_{I/O}$:	I/O CLOCK
WDx:	WRITE DDRx
RDx:	READ DDRx
WPx:	WRITE PORTx
RRx:	READ PORTx REGISTER
RPx:	READ PORTx PIN

Note: 1. WPx, WDx, RRx, RPx, and RDx are common to all pins within the same port. $clk_{I/O}$, SLEEP, and PUD are common to all ports.

AVR Ports – Configure input

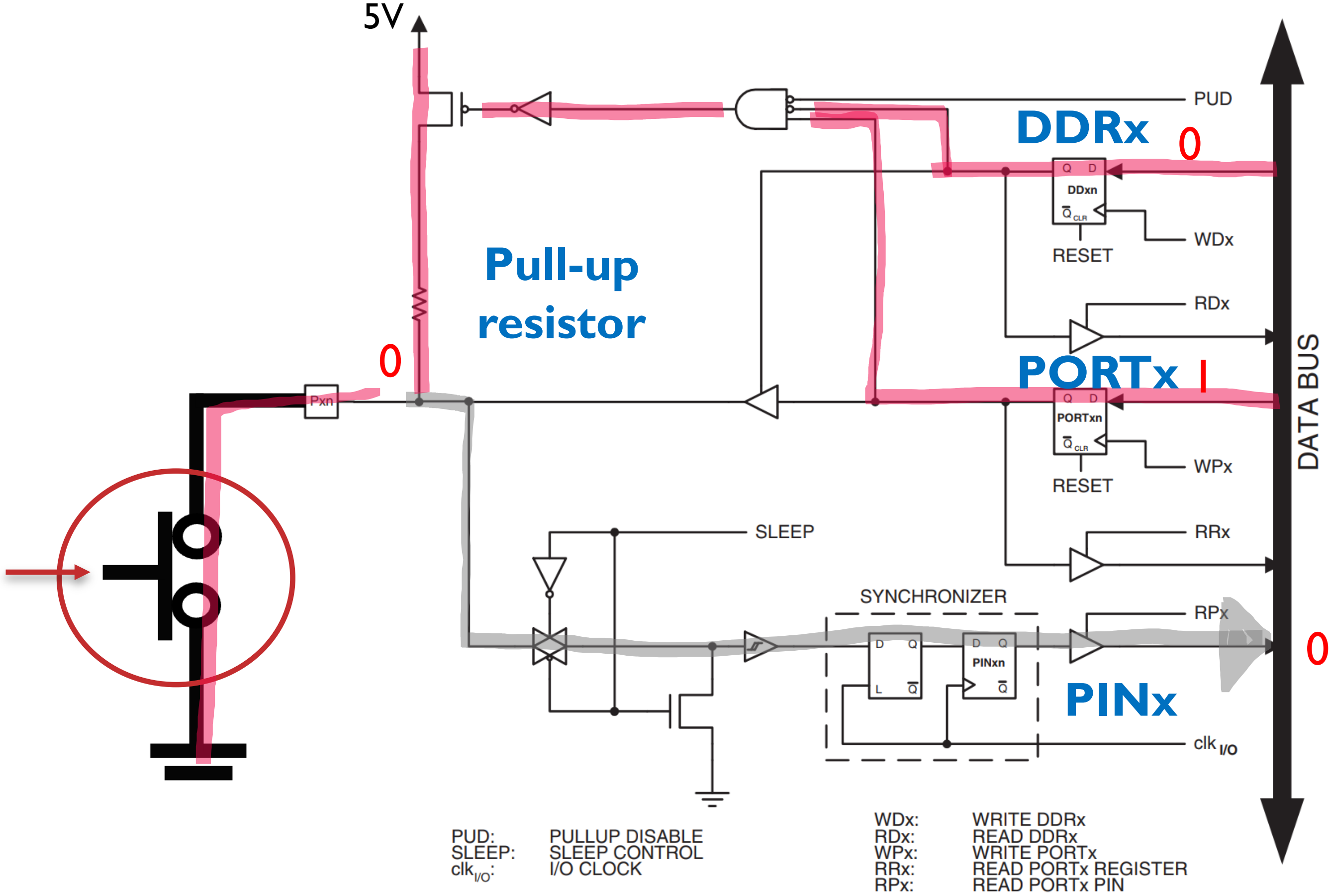
Figure 30. General Digital I/O[1]



Note: 1. WPx, WDx, RRx, RPx, and RDx are common to all pins within the same port. $clk_{I/O}$, SLEEP, and PUD are common to all ports.

AVR Ports – Configure input

Figure 30. General Digital I/O[1]



Note: 1. WPx, WDx, RRx, RPx, and RDx are common to all pins within the same port. $clk_{I/O}$, SLEEP, and PUD are common to all ports.

Switch Debouncing

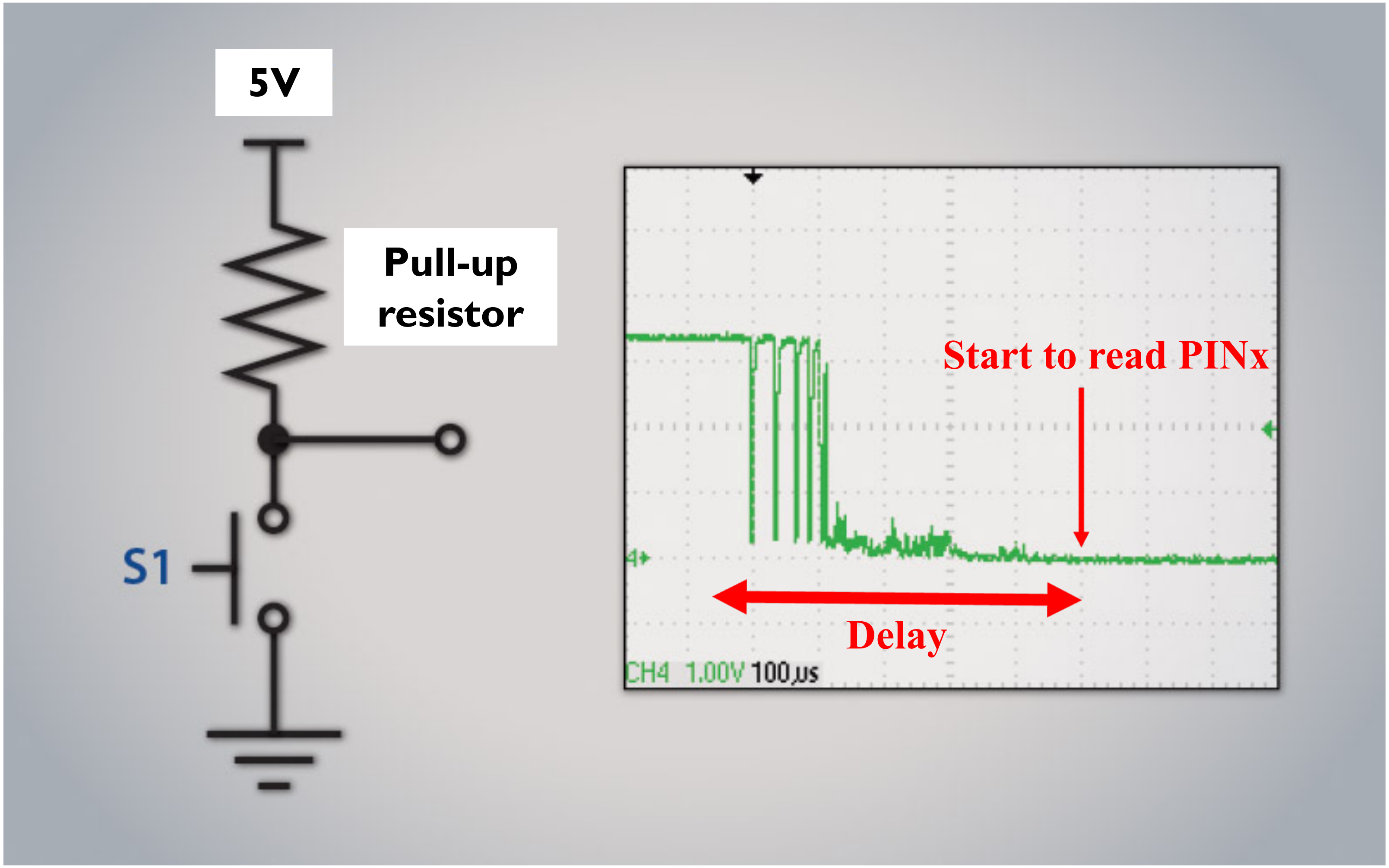


Image from google

C vs Assembly

- In C

```
DDRB =  0b11110000          // set 7-4th bits as outputs
PORTB = 0b01100000          // turn on LEDs connected to 5-6th bits
```

- In Assembly

```
LDI      mpr, 0b11110000
OUT      DDRB, mpr          ; set 7-4th bits as outputs
LDI      mpr, 0b01100000
OUT      PORTB, mpr         ; turn on LEDs connected to 5-6th bits
```

C vs Assembly

- In C

```
uint8_t mpr = PIND & 0b00110000;      // read and extract only 4-5th bit
If (mpr == 0b00100000)                // check if the right whisker is hit
{
    BotAction();                      // call BotAction
}
```

- In Assembly

```
IN        mpr,      PIND              ; read and
ANDI      mpr,      0b00110000        ; extract only 4-5th bit
CPI       mpr,      0b00100000        ; check if right whisker is hit
BRNE      NEXT                        ; if not, go to NEXT
RCALL     BotAction                   ; if yes, call BotAction
NEXT:
```

Check-off Lists

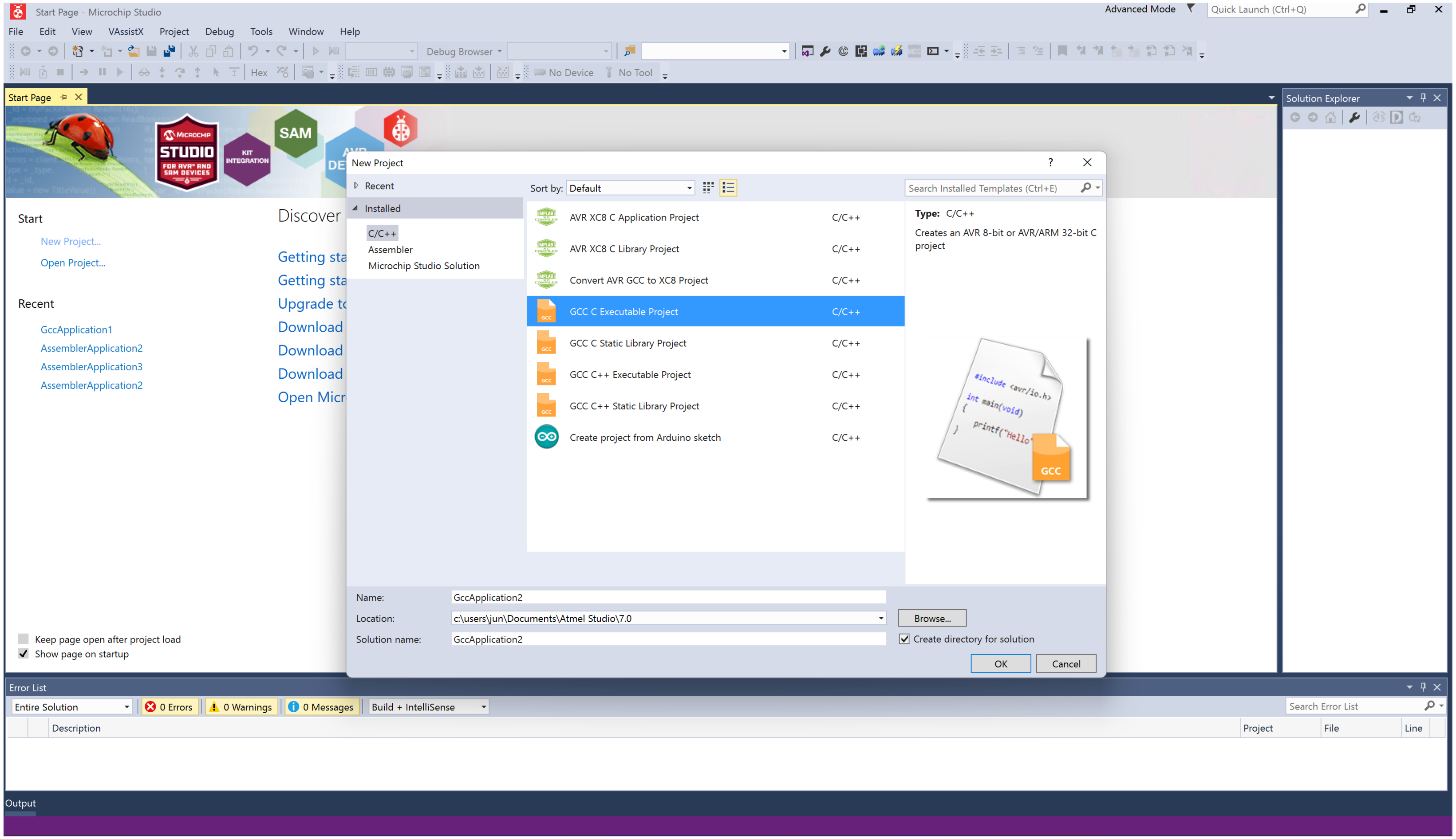
- Initialize Ports for input and output correctly.
- Detect whisker inputs correctly.
    - Left
    - Right
    - Both
- Correct Bumpbot behaviors accordingly to different triggering whiskers.
- Successfully translate the code file into a hex file.
- Successfully flash the code into the board.

Announcements

- Submit files by the due shown in Canvas.
  - Part 1 and 2 Source codes

Questions?

C Compilation for Windows



Sort by: Default

AVR XC8 C Application Project	C/C++
AVR XC8 C Library Project	C/C++
Convert AVR GCC to XC8 Project	C/C++
GCC C Executable Project	C/C++
GCC C Static Library Project	C/C++
GCC C++ Executable Project	C/C++
GCC C++ Static Library Project	C/C++
Create project from Arduino sketch	C/C++

Type: C/C++

Creates an AVR 8-bit or AVR/ARM 32-bit C project

Name: GccApplication2

Location: c:\users\jun\Documents\Atmel Studio\7.0

Solution name: GccApplication2

☑ Create directory for solution

C Compilation for Mac and Ubuntu users

- Install avr-gcc toolchain
- Download Makefile in the Lab webpage
- Open the Makefile with a text editor and set PRG variable to the source code file name excluding file extension(.c).
  - e.g., PRG = DanceBot