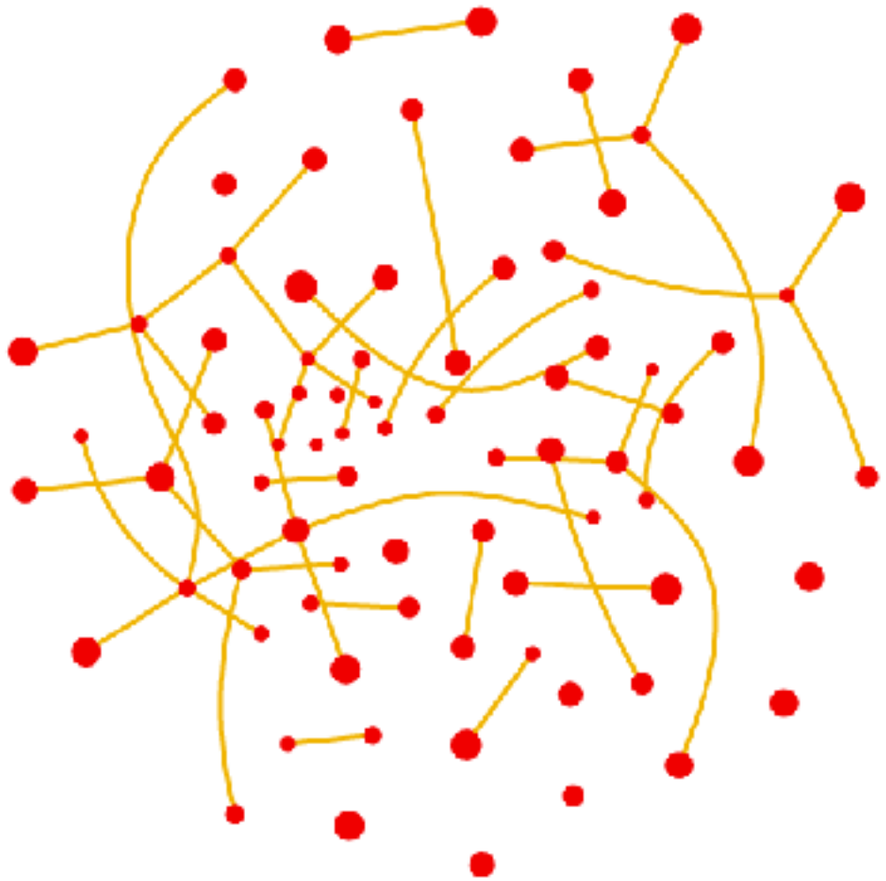


Optimize

Now you've got a functional prototype, you've got a week to optimize.

Optimize

There’s four things you can do easily:

Improve interactions

Improve code quality (QA the hell out of it)

Get off Breadboards/Shrink your circuit

EEPROM - Have it remember settings

Manage your Power

Report out on your user tests

What did you learn?

What do you need to improve?

What tweaks can you make?

What is most pressing?

Sufficient information design

Consistent and intuitive mapping

Match between system and real world

Visibility of state

Aesthetic and pleasing design

Useful and relevant information

Visibility of system status

User control and freedom

Easy transition to more in-depth information

Error prevention

"Peripherality" of display

Flexibility and efficiency of use

Optimize

There’s four things you can do easily:

~~Improve interactions~~

Improve code quality (QA the hell out of it)

Get off Breadboards/Shrink your circuit

EEPROM - Have it remember settings

Code Quality

1. One more code review. Yes.

2. Go back over the stateful example. Make sure it’s well organized

3. Q.A.

How to QA

Essentially it’s structured breaking.

Use the tasks you’ll perform during the demo or let people perform

Then perform them by NOT doing the expected things during the workflow

How to QA

QA’ing will give you a seamless demo. Spend time doing it.

The unexpected always happens. Try to preempt it.

Optimize

There’s four things you can do easily:

~~Improve interactions~~

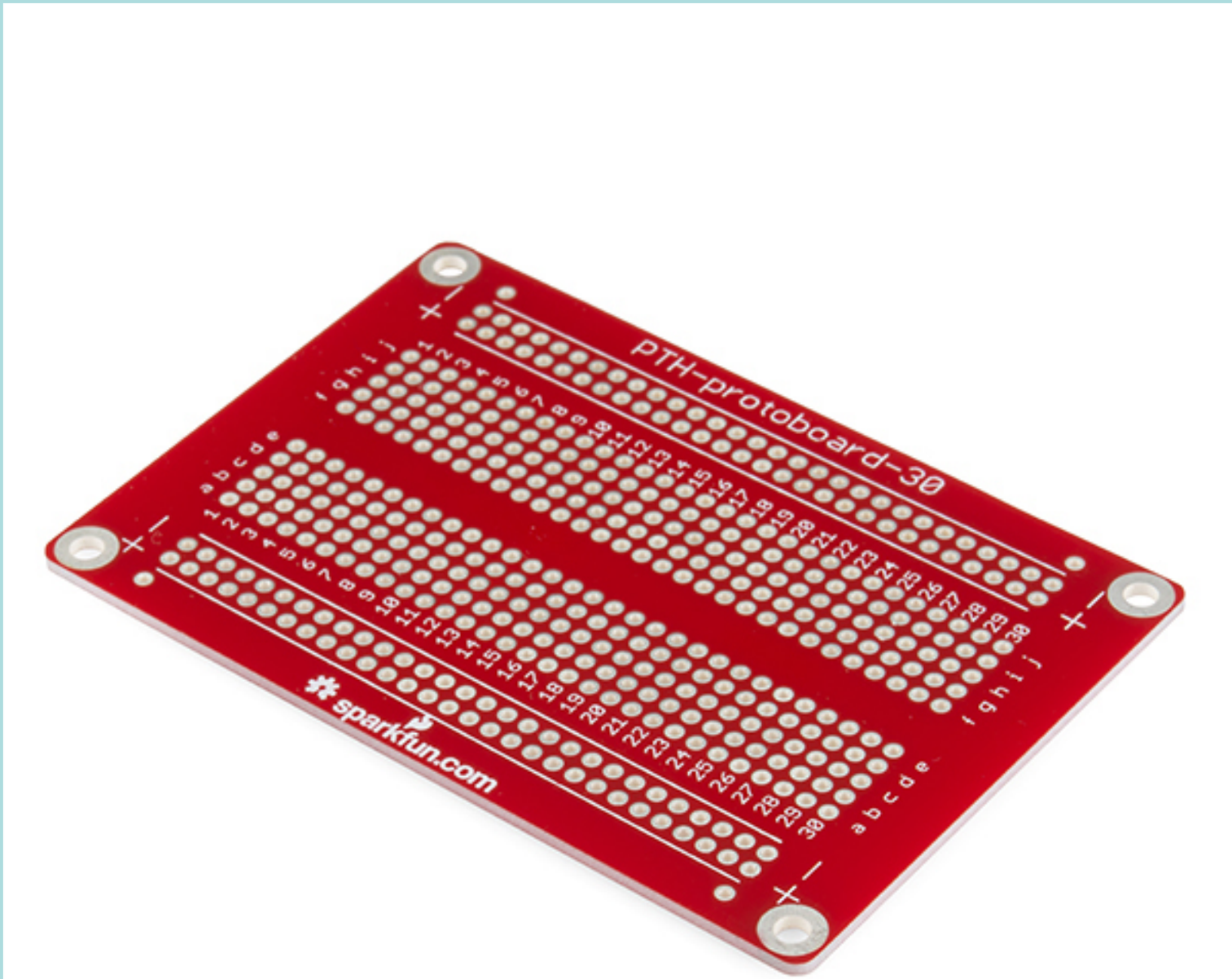
~~Improve code quality (QA the hell out of it)~~

Get off Breadboards/Shrink your circuit

EEPROM - Have it remember settings

Manage your Power

Moving to a Soldered Circuit



Soldered Boards

Before you begin:

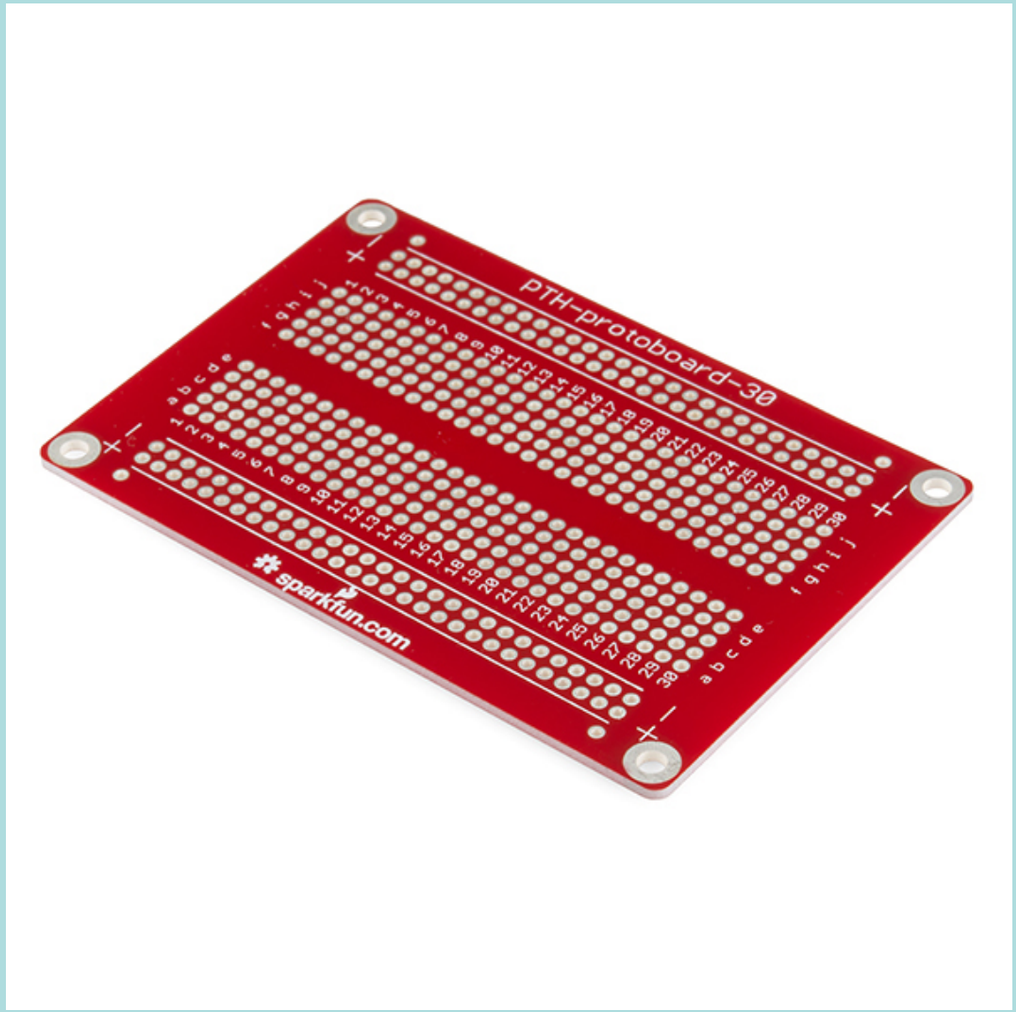
1. Make sure you've a working circuit

2. Make sure your circuit is final

3. Make sure you've optimized arrangement on the breadboard.

Soldered Boards

Perfboard (permanent solderable boards) map closely to breadboards. It’s easy to transfer



Plan your board.

1. Sketch on paper (or download a printable paper template)

2. Use Fritzing, diagram your circuit and check the layout.

Soldered Boards

Check

Double Check

Triple Check

Soldered Boards

Check

Double Check

Triple Check

P.S. Now is a good time to update your BoM!

Soldered Boards

Lay out your components on the protoboard.

Tape them in place

Flip them over and make sure the connections make sense

Solder.

Check for mistakes

[start over]

Plug in the Particle

[cross your fingers]

Soldered Boards

Rule of thumb:

Don’t solder expensive components to the board i.e. your Photon.

Instead use headers to fix them in place.

Moving to a Soldered Circuit

Image from: https://www.hackster.io/keithmitchell/controlling-my-desktop-computer-with-ifttt-and-photon-259281

Moving to a Soldered Circuit

http://www.evilmadscientist.com/2010/mailbag-moving-from-breadboard-to-protoboard/

Watch this:

https://www.youtube.com/watch?v=vMu1UjBxOiQ

Optimize

There’s four things you can do easily:

~~Improve interactions~~

~~Improve code quality (QA the hell out of it)~~

~~Get off Breadboards/Shrink your circuit~~

EEPROM + Connectivity: Have it remember settings

Manage your Power

EEPROM

"The EEPROM functions can be used to store small amounts of data in Flash that will persist even after the device resets after a deep sleep or is powered off."

EEPROM

```
// This function will write an object to the EEPROM.
// You can write single values like int and float
// or group multiple values together using struct.
```

```
EEPROM.put(int address, object)
```

```
// This function will retrieve an object from EEPROM.
// Use the same type of object from the put call.
```

```
EEPROM.get(int address, object)
```

EEPROM

It’s limited

2047 BYTES ONLY

int > uint > uint16_t > uint8_t

Structs

Allows variables to be grouped

For example:

- user preferences,
- location data,
- accelerometer info,

EEPROM

```
struct AccelerometerReading {
  float x;
  float y;
  float z;
};

// stores the reading
AccelerometerReading reading;

void loop()
{
   reading = { 20, 403, 234 };
   // or
   reading.x = analogRead( accelXPin );
    Serial.println( reading.x );
}
```

EEPROM

Storing Settings

```
// EEPROM Struct for storing application settings
int eeprom_address = 0;

struct AppSettings {
  uint8_t version;
  int timeBetweenPublish;
  char usersName [256];
};

AppSettings settings; // stores the main settings
```

EEPROM

```
void setup() {
  Serial.begin(9600);               // opens serial port for LCD shield

  Serial.println( "Loading settings. ");
   loadSettingsFromMemory();

   Particle.function( "setUserName", updateUsersName );
  Particle.function( "setPubTime", updateTimeBetweenPublish );


}
```

EEPROM

```
void loadSettingsFromMemory()
{
  EEPROM.get( eeprom_address, settings );
  if( settings.version != 1) {
    // EEPROM was empty -> initialize with defaults
    Serial.println( "Setting Defaults" );

    settings = getDefaultSettings();
  }

  Serial.println( "Settings Loaded" );
  Serial.print( "version = " );
  Serial.println( settings.version );

}
```

EEPROM

```
struct AppSettings getDefaultSettings(){
  AppSettings defaults = { 1, 1000, “unknown" };
  return defaults;

}
```

EEPROM

```
int updateTimeBetweenPublish( String command )
{
  Serial.print( "updateTimeBetweenPublish received: " );
  Serial.println( command );

  int timeInSeconds = command.toInt();

  if( timeInSeconds > 0 )
  {
     settings.timeBetweenPublish = timeInSeconds * 1000;
    saveSettingsToMemory();

    return 0;
  }
  return -1;
}
```

Structs

Provided a detailed guide and full fledged tutorial for a digital Magic 8-ball in repo

Optimize

There’s four things you can do easily:

~~Improve interactions~~

~~Improve code quality (QA the hell out of it)~~

~~Get off Breadboards/Shrink your circuit~~

~~EEPROM - Have it remember settings~~

Manage your Power

Power

Passing the buck

Particle

Power Management for IoT Devices