

Smart Saw

SD-May32: Jace Fedler, Mitchell Kistner, Ethan Bauman, Austin Mackedanz, Patrick Pham, David Kruze, Lance Longhorn

What is Smart Saw?

- Upgrading client's existing Branch Saw to wireless
- Reduces
 - Dangers of tree trimming
 - Cost of professional tree trimmers
- 3 Main Components
 - Electric chainsaw
 - Main rotation bar
 - Claw
- 3 main steps of operation
 - Attach claw to desired branch
 - Power the electric chainsaw
 - Rotate main rotation bar





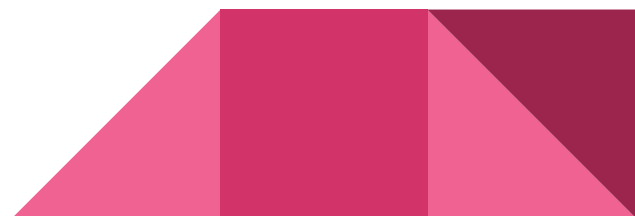
Electric Chainsaw

12V Brushed Motor



Main Rotation Bar

360 Degree Rotation



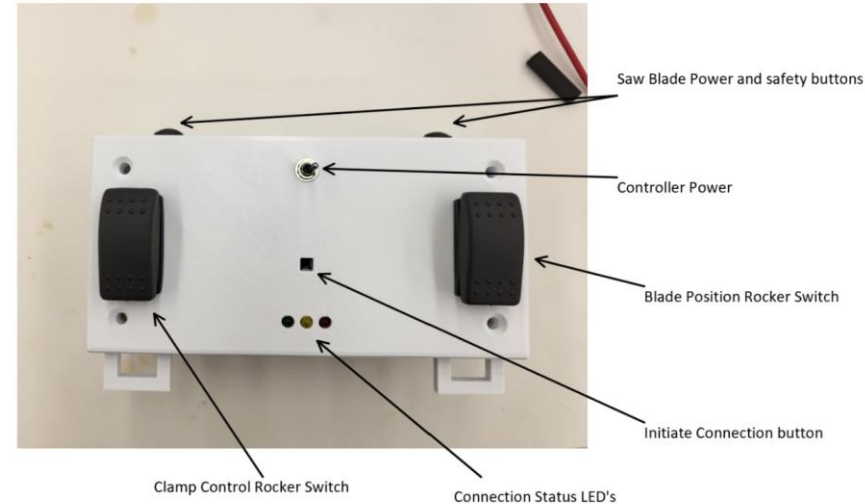
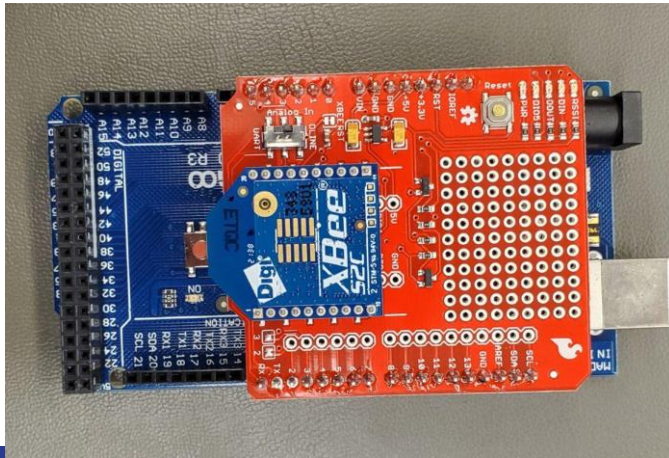


Claw

Strong enough to clamp onto trees,
with studs to aid in grip

What Makes Smart Saw Different?

- Complete wireless control of Saw, Claw, and Rotation Bar
 - Using two Arduino and Xbee transceivers
- Creation of a controller
- Housing for the Saw mounted Arduino



Architecture Overview

- Subsystems
 - Saw
 - Arduino
 - XBEE
 - Relays to Saw Components
 - Toggle switch for communication power
 - Controller
 - Arduino
 - XBEE
 - Rocker switch for bidirectional control
 - Button for safety
 - Button for run

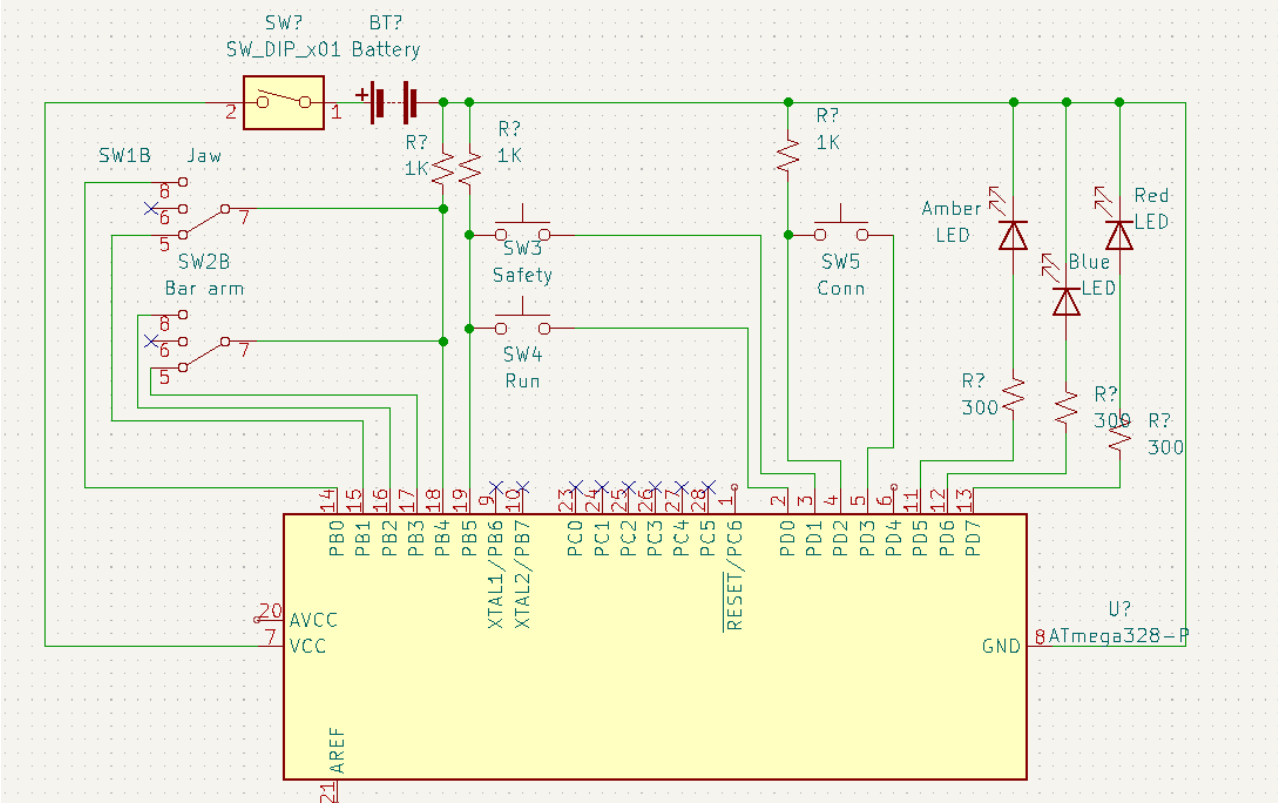


Hardware/Electrical

- Saw
 - Relays
 - Box for added electronics
- Controller
 - Box to act as the controller
 - Switches and buttons
 - Status LEDs
- Both
 - Xbee
 - Arduino/Arduino Power



Controller Schematic



Software/Controller

- Data packet containing bits with information being sent between the two arduinos
- Communication detection the the two arduinos
- Sequence to start saw blade.
- Error detection and reset
- ID numbers for controller and saw
- Proper handshake between the two arduinos



Data Transmission Code

```
if (XBee.available())//Read incoming Data from other controller
{ // If data comes in from XBee, send it out to serial monitor
  cTempDataCollect = XBee.read();
  if(cTempDataCollect == '*')
  {
    iDataInCounter = 0;
    bReadComplete = LOW;

    while((bReadComplete == LOW) && (iDataInCounter <= 5))
    {
      if (XBee.available())
      {
        cTempDataCollect = XBee.read();
        //Serial.write(cTempDataCollect);
        if(cTempDataCollect == '!')
        {
          bReadComplete = HIGH;
          uiDataIn = DataToByte(arCharCollect);
          arCharCollect[0] = ' ';
          arCharCollect[1] = ' ';
          arCharCollect[2] = ' ';
          arCharCollect[3] = ' ';
          arCharCollect[4] = ' ';
          iXbeeCounter = 0;
        }
        else
        {
          arCharCollect[iDataInCounter] = cTempDataCollect;
          iDataInCounter = iDataInCounter + 1;
        }
      }
    }
  }
}
//end of data read loop
```

```
unsigned int DataToByte (char DataIn[5])//Method to take char inputs form data input and convert it to byte for use within program
{
  unsigned int DataOut = 0;
  int counter = 0;

  if(DataIn[4] != ' ')
  {
    DataOut = DataOut + ((DataIn[4] - 48) * power(10,counter));
    counter = counter + 1;
  }
  if(DataIn[3] != ' ')
  {
    DataOut = DataOut + ((DataIn[3] - 48) * power(10,counter));
    counter = counter + 1;
  }
  if(DataIn[2] != ' ')
  {
    DataOut = DataOut + ((DataIn[2] - 48) * power(10,counter));
    counter = counter + 1;
  }
  if(DataIn[1] != ' ')
  {
    DataOut = DataOut + ((DataIn[1] - 48) * power(10,counter));
    counter = counter + 1;
  }
  if(DataIn[0] != ' ')
  {
    DataOut = DataOut + ((DataIn[0] - 48) * power(10,counter));
  }

  Serial.println(DataOut);

  return DataOut;
}

int power (int number, int exponent)
{
  int output = 1;
  while (exponent > 0)
  {
    output = output * number;
    exponent--;
  }
  return output;
}
```

```
//Output Data signal through xbee
if(iXbeeCounter >= 0)
{
  XBee.print('*');
  XBee.print(uiDataOut);
  XBee.print('!');
  iXbeeCounter = -100;
}
iXbeeCounter ++;
```

Communication Data Sheet

Controller -> Saw : Byte 1								
BIT #	16	15	14	13	12	11	10	9
BIT Use	Matched Controller ID #	Matched Controller ID #	Matched Controller ID #	Matched Controller ID #	Matched Controller ID #	Matched Controller ID #	Matched Controller ID #	Matched Controller ID #
When Value 0								
When Value 1								

Controller -> Saw : Byte 0								
BIT #	8	7	6	5	4	3	2	1
BIT Use	Heartbeat Signal	Error	Saw Blade	Clamp Open	Clamp Close	Blade Rotate Forward	Blade Rotate Reverse	Error Reset
When Value 0	HB Off	No Active Error	Blade Off	No Movement/Close	No Movement/Open	No Movement/Reverse	No Movement/Forward	No Reset Of Saw Error
When Value 1	HB On	Active Controller Error	Blade On	Open Clamp	Close Clamp	Clockwise Motion	Counter Clockwise Motion	Reset The Saw Error

Saw -> Controller : Byte 1								
BIT #	16	15	14	13	12	11	10	9
BIT Use	Saw ID #	Saw ID #	Saw ID #	Saw ID #	Saw ID #	Saw ID #	Saw ID #	Saw ID #
When Value 0								
When Value 1								

Saw -> Controller : Byte 0								
BIT #	8	7	6	5	4	3	2	1
BIT Use	Heartbeat Signal	Error	Reset Ack	Blade On				
When Value 0	HB Off	No Active Error	Reset Bit not Recieved	Blade Is Off				
When Value 1	HB On	Active Saw Error	Reset Bit Recieved	Blade Is Running				

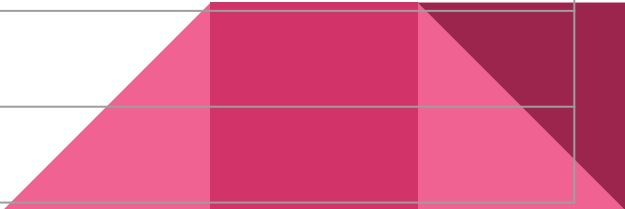
Work Accomplishments

- Implemented wireless control of arm bar, clamp, and motor
 - Added relays to allow the controls to interact easier with the Arduino
 - LED indicators to indicate current status
- 3D printed controller and brain box of the saw
 - Multiple iterations to better suit our needs
 - Added threaded inserts to hold device together
- Established wireless communication between Arduinos
 - Created Data Packet To Send Current Data Between The Two Arduinos
 - Created ID to Link Controller with a specific saw
 - Proper handshake between the saw and controller
- No one was injured while working on project



Key Contributions

Name	Team	Responsibility
Mitchell Kistner	Hardware	Designed controller schematic, troubleshooting, communication with client
David Kruze	Hardware	Responsible for soldering all connections for the controller box and saw control box.
Lance Longhorn	Hardware	Responsible for 3D model design and 3D printing of controller and saw control box.
Austin Mackedanz	Controls	Setting up xbee boards for communication. And help with testing
Ethan Bauman	Controls	Responsible for Saw and Controller programs and data communication between the two devices. Designed Electrical layout for the Saw
Patrick Pham	Hardware	Testing and documentation
Jace Fedler	Controls	Documentation of progress and testing



Challenges and Solutions

- Parts needed to have large scale production in mind
 - Had to buy from only reputable suppliers
- Saw function
 - Lost motor function in Early April
 - Bad diagnosis on issue
- Ability to charge batteries
 - Needed charger with lipo connection capability



Future Modifications/Prospects

- Auto Cut
 - No need to move the bar arm yourself
 - Work its way through the branch itself
- 1 button run
 - Client wanted the saw to orient itself and figure out the most efficient cut
 - Hands off approach
- Power optimization
 - Adjust current for blade to save power



Conclusion

We added the implementation requested by our client that we could after the motor controller was broken. The code in place should also work when the motor controller is hooked back up giving the controller the ability to also run the blade.

