USO DE RESIDUOS DE APARATOS ELECTRICOS Y ELECTRONICOS EN LA CONSTRUCCION DE UNA IMPRESORA 3D

ARIAS GUTIÉRREZ WILSON ALFONSO

LONDOÑO BUITRAGO FERNANDO ALONSO

MUÑOZ GONZÁLEZ FABIÁN LEANDRO

INSTITUCIÓN UNIVERSITARIA PASCUAL BRAVO

FACULTAD DE INGENIERÍA

TECNOLOGÍA EN SISTEMAS MECATRONICOS

MEDELLÍN

2017

USO DE RESIDUOS DE APARATOS ELECTRICOS Y ELECTRONICOS EN LA CONSTRUCCION DE UNA IMPRESORA 3D

ARIAS GUTIÉRREZ WILSON ALFONSO

LONDOÑO BUITRAGO FERNANDO ALONSO

MUÑOZ GONZÁLEZ FABIÁN LEANDRO

TRABAJO DE GRADO PARA OPTAR AL TITULO DE TECNOLOGO EN SISTEMAS MECATRONICOS

ASESOR

MARIA VICTORIA HERRERA DEDERLÉ

INGENIERA ELECTRONICA

MAGISTER EN DIRECCION ESTRATEGICA EN TELECOMUNICACIONES

INSTITUCIÓN UNIVERSITARIA PASCUAL BRAVO

FACULTAD DE INGENIERÍA

TECNOLOGÍA EN SISTEMAS MECATRONICOS

MEDELLÍN

2017

Contenido

Lista De Figuras

Lista De Anexos

RESUMEN

USO DE RESIDUOS DE APARATOS ELECTRICOS Y ELECTRONICOS EN LA CONSTRUCCION DE UNA IMPRESORA 3D

ARIAS GUTIÉRREZ WILSON ALFONSO

LONDOÑO BUITRAGO FERNANDO ALONSO

MUÑOZ GONZÁLEZ FABIÁN LEANDRO

El trabajo escrito *USO DE RESIDUOS DE APARATOS ELECTRICOS Y ELECTRONICOS* EN LA *CONSTRUCCION* DE UNA *IMPRESORA 3D*, da a conocer al lector si es posible la creación de una impresora 3d utilizando residuos de aparatos electrónicos y electrónicos desechados por personas a las que se le dañaron o ya no le encontraron más uso, y entendiendo que muchas cosas de la impresora se deben comprar en almacenes de electrónica, como la parte electrónica, la materia prima que es el PLA, el MK8, extrusor, etc.

La parte *mecánica de movimiento*, como motores, poleas, sistema de guías, carcasa de la impresora, y la parte de de la alimentación como lo es la fuente de alimentación es más posible de construir con reciclaje. Este trabajo muestra de una manera muy clara y practica el porqué de construir una impresora de esta manera, para que construirla?, como construirla?, el objetivo general y objetivos específicos, definición de cada elemento utilizado en la construcción de dicho proyecto, un paso a paso con imágenes caseras de cómo fue la construcción de la impresora, resultados de la construcción y puesta en marcha, recomendaciones a la hora de construir una impresora similar, *presupuesto* utilizado para el ensamble que es mucho menos del presupuesto utilizado si se hubiera construido

completamente con partes compradas en un almacen de electronica y conclusiones de lo que se vio y se analizó a la hora de la construcción y puesta en marcha.

ABSTRACT

USO DE RESIDUOS DE APARATOS ELECTRICOS Y ELECTRONICOS EN LA CONSTRUCCION DE UNA IMPRESORA 3D

ARIAS GUTIÉRREZ WILSON ALFONSO

LONDOÑO BUITRAGO FERNANDO ALONSO

MUÑOZ GONZÁLEZ FABIÁN LEANDRO

The written work *USE OF WASTE OF ELECTRICAL AND ELECTRONIC APPLIANCES* IN THE *CONSTRUCTION* OF A *3D PRINTER*, tells the reader if it is possible to create a 3d printer using waste electronic and electronic devices discarded by people who were damaged or They no longer found use, And understanding that many things of the printer should be bought in electronics stores, such as electronic part, raw material which is PLA, MK8, extruder, etc.

The *mechanical part of movement* such as motors, pulleys, guide system, printer housing, and the part of the power supply as it is the power supply is most possible to build with recycling. This work shows in a very clear and practical way why to build a printer in this way, to build it ?, how to build it ?, the general objective and specific objectives, definition of each element used in the construction of such project, A step by step with home-made images of how the printer was built, construction results and start-up,

recommendations when building a similar printer, *Budget* used for assembly that is much less than the budget used if it had been built completely with parts purchased in an electronic storage and conclusions of what was seen and analyzed at the time of construction and start-up.

Glosario

Acoples Rápidos: sistema de conexión de mangueras o tubos flexibles que se pueden conectar y desconectar.

Arduino: computador a pequeña escala con el que se puede hacer cualquier clase de proyecto.

Drivers: mediante ordenes análogas del arduino el driver las convierte en señales eléctricas y las envía hacia cada bobina de los motores para ir dando movimiento por paso, este driver cuenta con un potenciómetro para calibrar la corriente de lo que va a alimentar.

Extrusor: sistema que funciona como resistencia, calentando el material y expulsándolo derretido por una boquilla.

Finales De Carrera: pulsadores con conexiones NA, NC y COM, para hacer la conexión que desee y se utilizan para determinar el paso de un objeto o el final o principio del movimiento de determinado objeto.

Marlín: programa de impresoras 3d que se descarga en el arduino y maneja todo el sistema.

Motor Pasó A Paso: capaz de girar una cantidad de grados (paso o medio paso) dependiendo de sus entradas de control, precisión y repetitividad en cuanto al posicionamiento.

Mk8: sistema mecánico acoplado a un motor que hace mover lentamente el PLA hacia el extrusor.

Nema 17: motores paso a paso dipolares de buen torque utilizados comúnmente en impresoras 3d y cncs.

Pla: material más común en a impresión 3d, material termoplástico, La materia prima destacable del **PLA** es el **maíz** (material ecológico).

Pronterface: programa de computador utilizado para darle puesta a punto a la impresora 3d.

Ptfe: polímero similar al polietileno.

RAEE: residuos de aparatos eléctricos y electrónicos.

Ramps 1.4: placa electrónica de potencia que traduce las órdenes digitales de un ordenado, en órdenes por pasos de los motores.

Introducción

El trabajo USO DE RESIDUOS DE APARATOS ELECTRICOS Y ELECTRONICOS EN LA CONSTRUCCION DE UNA IMPRESORA 3D , es realizado con el fin de saber si es posible la construcción de una impresora 3d con residuos de aparatos eléctricos y electrónicos, en este caso partes de computadores viejos, partes de impresoras de papel en mal estado y la mayor parte de cosas reciclables que sean posible en la construcción de la impresora, teniendo en cuenta que varios de los elementos necesarios para la construcción del proyecto, se deben comprar en un almacén de electrónica ya que estos elementos no son fáciles de encontrar en residuos de aparatos viejos.

La parte física en la que están los movimientos de la impresora, son las que se van a ensamblar de una forma casera utilizando guías de cartuchos de tintas de impresoras de papel recicladas, los motores de los ejes, la superficie donde se va a imprimir y la fuente de alimentación de la impresora en la que se va a utilizar una fuente de alimentación de computador reciclada.

Esta idea busca incentivar la creatividad y utilizar los residuos de aparatos eléctricos y electrónicos para disminuir el impacto que ellos ocasionan en el medio ambiente y en el mundo.

1. Planteamiento del problema

1.1 Descripción:

La producción mundial de aparatos electrónicos y, en particular de tecnologías de la información y las comunicaciones (TIC) se enfrenta a la mayor expansión industrial de la historia: según cifras de la Organización para la Cooperación y el Desarrollo Económico (OCDE), el comercio mundial de las TIC alcanzó el 7,7% del producto mundial bruto en 2004, la mayor parte procedente de China [1]. Se estima que en el 2006, 230 millones de computadores y mil millones de teléfonos celulares se vendieron en todo el mundo, lo que corresponde a 5'848.000 toneladas. Como consecuencia, los residuos de aparatos eléctricos y electrónicos son, por mucho, el componente de los residuos de más rápido crecimiento. Según el PNUMA la generación de RAEE en los países en vía de desarrollo se triplicará hacia el año 2010.

(Ministerio del Medio Ambiente, 2010)

En Europa los residuos electrónicos están experimentando un crecimiento del 3 al 5% al año, casi 3 veces más rápido que el total de los residuos generados1. La cantidad actual de RAEE generados en los 27 países miembros de la Unión Europea (EU27) se estima en 8,7 millones de toneladas al año, mientras que la cantidad recogida y reciclada se estima en sólo 2,1 millones de toneladas o el 25%. Esta estimación incluye todas las categorías de los desechos electrónicos definidas por la legislación europea.

(Ministerio del Medio Ambiente, 2010)

En los EE.UU., menos del 20% de las categorías como televisores, computadores y periféricos incluidos los teléfonos móviles, fueron separados de las otras corrientes de desechos para "tratamiento y recuperación posterior". Esta cifra incluye parte de la exportación de desechos electrónicos a países como India y China. El resto es incinerado, enviado a los rellenos, almacenado, reutilizado o exportado.

(Ministerio del Medio Ambiente, 2010)

En 1994 se estimaba que aproximadamente 20 millones de computadores personales (PC), cerca de 7 millones de toneladas, quedaron obsoletos. Hacia 2004, esa cifra se había

incrementado a más de 100 millones de PC. En cifras totales, cerca de 500 millones de PC alcanzaron el fin de su vida útil entre 1994 y 2004.En total, el crecimiento de productos electrónicos desechados a escala mundial se calcula entre 20 y 50 millones de toneladas generados cada año.

(Ministerio del Medio Ambiente, 2010)

Las impresoras 3D son máquinas que pueden crear casi cualquier diseño que se quiera volver realidad, siendo de gran utilidad en los temas de tecnología e innovación, ayudando en temas de estudios universitarios donde es posible innovar en la construcción de nuevos prototipos de aparatos, como es la carrera de mecatrónica. De esta manera puede abarcar la mayoría de campos donde se desarrolle tecnología, como por ejemplo en la medicina para crear prótesis, en la automoción para crear prototipos, en la industria para crear carcasas de cualquier tipo de máquinas o aparatos electrónicos; en todo tipo de utilidades donde sea necesario en uso de plásticos que es un elemento muy utilizado actualmente.

El proyecto busca darle uso a los residuos de aparatos eléctricos y electrónicos para demostrar que con creatividad se le puede dar uso a residuos desechados que no tienen ningún valor y convertirlos en proyectos tecnológicos de la nueva era, reduciendo costos, ayudando a pensar más en la forma de diseñar y no comprar todos accesorios en un almacén de electrónica y llegar solo a armar puesto que no se está siendo innovador y creativo, y a su vez ayudando en un tema muy importante en la actualidad, que es aportar en la reducción del impacto ambiental que generan los residuos de aparatos eléctricos y electrónicos (RAEE), tanto en cada país, como en el mundo entero.

1.2 Formulación:

¿Es posible dar un uso a residuos de aparatos eléctricos y electrónicos para el diseño de una impresora 3D?

2. Justificación

En América Latina, el reciclaje formal de los desechos electrónicos, que en su mayoría se limita a un desensamble profesional, es una actividad bastante nueva. En países como Chile, Argentina, Perú, Colombia y Brasil, empresas tradicionales de reciclaje de metales han descubierto el mercado de reciclaje de los RAEE, sin embargo, las cantidades recicladas están todavía en un nivel modesto, ya que ni el marco político, ni la infraestructura logística permiten mayores cantidades. La mayoría de estas empresas no ofrecen un servicio completo, ya que se concentran básicamente en los componentes valiosos, como las tarjetas de circuito impreso, descuidando la disposición adecuada de otros componentes como los tubos de rayos catódicos (TRC) que no tienen un valor económico, pero representan un riesgo para la salud y el medio ambiente.

(Ministerio del Medio Ambiente, 2010)

En Chile, el reciclaje formal de los RAEE alcanza sólo un 1,5 a 3% de las cantidades generadas [6], una cifra que probablemente es similar o incluso inferior en los demás países de la región. La mayoría de las empresas se concentran en la prestación de servicios a grandes empresas nacionales e internacionales basándose en un enfoque empresa a empresa (B2B: bussiness to bussiness), mientras que el sector informal está tratando de beneficiarse de los componentes valiosos de los residuos procedentes de hogares particulares.

(Ministerio del Medio Ambiente, 2010)

Se estima que en los países de América Latina se están generando aproximadamente 120.000 toneladas al año, una cantidad que se triplicará hacia el 2015. A la pregunta ¿si el tema de los residuos electrónicos ya ha alcanzado una masa crítica en América Latina y el Caribe? Ripley responde lo siguiente: “El potencial de LAC para generar cantidades

considerables de RAEE ha crecido drásticamente en los últimos años. Las ventas de computadores personales y teléfonos celulares se han disparado.

(Ministerio del Medio Ambiente, 2010)

Pero el problema va más allá de computadores y celulares. Una amplia gama de equipos digitales que en los Estados Unidos y Europa ya se dan por sentados, apenas empezaron a conquistar los mercados de LAC. Además se puede observar que los usuarios latinoamericanos ya no se contentan con comprar los modelos de ayer". Las mismas tendencias también se pueden observar en Colombia (Ilustración 1). Las ventas de equipos eléctricos y electrónicos se han disparado en los últimos años, y en poco tiempo estos aparatos serán descartados por sus usuarios convirtiéndose en residuos.

(Ministerio del Medio Ambiente, 2010)

Gran parte de los inventos creados por el hombre están compuestos de diferentes tipos de materiales, los metales y plásticos son dos de los más comunes que encontramos normalmente en una maquina o aparato, esto quiere decir que la creación de elementos plásticos que puede hacer una impresora 3d, tiene muchísimos campos de aplicación de una manera más económica y práctica.

La impresión 3d, la domótica, los drones, las maquinas cnc, la fibra óptica, entre otras invenciones modernas; son tecnologías que se han creado en la actualidad y han cambiado la manera de crear productos, han mejorado las condiciones de vida de muchas persona, han reemplazado la manera artesanal y algunas veces la manera complicada en la que se hacían algún tipo de producto, han cambiado la forma de grabar videos, de mejorar las comunicaciones, de producir estructuras en madera, de producir productos para todo tipo de aparatos; como toda invención que cambia de alguna u otro manera la vida de los seres humanos y muchos seres vivos.

Así como se crean nuevos productos y tecnologías, crece la facilidad y la economía de encontrar los materiales para ensamblarlos, videos en internet en los que muestran cómo se hacen y se arman, almacenes donde se puede comprar todos los accesorios para hacer una de estas máquinas, productos que podemos encontrar fácilmente en las ciudades y que anteriormente era más difícil de conseguir, ayudando en la educación de una forma

didáctica e incentivar la creatividad, la innovación con talleres y cursos donde los jóvenes pueden interactuar con todos estos tipos de tecnologías que son sorprendentes pero al interactuar con ellas y saber su funcionamiento se hacen un poco más fáciles de entender.

El desarrollo de la ciencia y la tecnología trae de por medio el aumento de materiales reciclables que si no son bien utilizados afectan el medio ambiente, por esta razón la importancia de reciclar todo tipo de productos y más aún, los aparatos electrónicos bajo unos lineamientos técnicos ya definidos para el manejo de residuos de aparatos eléctricos y electrónicos (RAEE), que están construidos de un sin número de elementos que son hechos de materiales químicos y que son muy nocivos para el medio ambiente, elementos como resistencias, microchips, diodos, trasformadores, reguladores, bobinas, capacitores, etc.; y que la mayoría de veces echan a la basura buenos. Normalmente un aparato electrónico está compuesto de muchos sistemas electrónicos que en conjunto hacen que una maquina desempeñe una función y la cantidad de elementos que la componen hacen más grande las posibilidades de que el aparato falle por el daño de alguno de esos elementos, pero eso es lo que pasa normalmente que echan a la basura una impresora de papel, un equipo de sonido, un computador con muy pocas cosas malas y la mayoría de las otras en buen estado, y la reutilización de estos elementos electrónicos tiene muchas aplicaciones en cualquier cosa que se desee hacer y más económico y con creatividad y pasión no hay límites.

Estos Lineamientos técnicos son ya definidos de la forma de reciclar ese tipo de aparatos eléctricos y electrónicos regidos por el ministerio de ambiente, vivienda y desarrollo territorial, el cual dicta la manera de recolección, manejo, almacenamiento, transporte y logística, reuso; entre otros, de este tipo de aparatos eléctricos y electrónicos que es necesario reciclar para ayudar en el impacto que generan al medio ambiente y a la población mundial.

3. Objetivos

3.1 Objetivo general:

Implementar una estructura de una impresora 3D funcional con uso de tecnología arduino y residuos de aparatos eléctricos y electrónicos para la construcción de la misma.

3.2 Objetivos específicos

1. Aplicar los conocimientos adquiridos en la tecnología en sistemas mecatrónicas para el desarrollo de nuevas tecnologías.
2. Construcción de estructura 3D con algunos residuos de aparatos eléctricos y electrónicos (RAEE), para la fabricación de equipos u herramientas de uso cotidiano en la mecatrónica.
3. Ensamblar la estructura física, conexiones del sistema electrónico, descarga de programas, ensamble general y puesta en marcha del proyecto, mediante la investigación del funcionamiento de las impresoras 3d.
4. Aportar al cuidado del medio ambiente utilizando algunos RAEE en la construcción de herramientas de uso necesario en la mecatrónica.
5. Proyectar a la comunidad el conocimiento de las nuevas tecnologías usadas para la creación de prototipos y máquinas.

4. Marco teórico

Los aparatos eléctricos y electrónicos (RAEE) están compuestos de cientos de materiales diferentes, tanto valiosos como potencialmente peligrosos. Oro, plata, paladio y cobre son algunos de los materiales valiosos que se pueden recuperar de los RAEE; plomo, cadmio, mercurio y arsénico son algunos de los componentes peligrosos que pueden estar presentes en los equipos eléctricos y electrónicos en desuso, lo cual va a depender del tipo de tecnología, país de origen y del fabricante, estos compuestos se pueden liberar al medio ambiente durante el desensamble de los mismos. Uno de los ejemplos más relevantes en cuanto al contenido de compuestos peligrosos es el plomo el cual estaba presente en la soldadura de muchos equipos, hoy en día en el mercado se ofrece equipos libres de soldadura de plomo.

(Ministerio del Medio Ambiente, 2010)

Estas características tan particulares de reunir, por ejemplo en un volumen tan pequeño como en el de un teléfono móvil, materiales de alto valor junto con elementos potencialmente peligrosos, son una de las causas de los impactos negativos que se generan al medio ambiente cuando se disponen en rellenos sanitarios, se botan a los suelos o cuerpos de agua o se realiza el desensamble inadecuado de estos residuos, ya que en algunos países en vía de desarrollo existe una fuerte lucha por los materiales de alto valor económico, en combinación con un fuerte desconocimiento de lo que se debe manejar de manera adecuada

(Ministerio del Medio Ambiente, 2010)

Que es una impresora 3D?

Una impresora 3D es una máquina capaz de realizar réplicas de diseños en 3D, creando piezas o maquetas volumétricas a partir de un diseño hecho por ordenador, descargado de internet o recogido a partir de un escáner 3D. Surgen con la idea de convertir archivos de 2D en prototipos reales o 3D. Comúnmente se ha utilizado en la prefabricación de piezas o componentes, en sectores como la arquitectura y el diseño industrial. En la actualidad se está extendiendo su uso en la fabricación de todo tipo de objetos, modelos

para vaciado, piezas complicadas, alimentos, prótesis médicas (ya que la impresión 3D permite adaptar cada pieza fabricada a las características exactas de cada paciente), etc.

(Wikipedia c. d., 2017)

La impresión 3D en el sentido original del término se refiere a los procesos en los que secuencialmente se acumula material en una cama o plataforma por diferentes métodos de fabricación, tales como polimerización, inyección de aporte, inyección de aglutinante, extrusión de material, cama de polvo, laminación de metal, depósito metálico.

(Wikipedia c. d., 2017)

Cada vez está más presente la tecnología en las aulas, las impresoras 3D son unas de las grandes apuestas para los próximos años para el sector educación como apoyo en determinadas asignaturas gracias a la posibilidad de materialización de un concepto estudiado en un objeto real. En España, la Comunidad de Madrid ha sido pionera al anunciar que dotará a más de 300 Institutos de Educación Secundaria con una impresora 3D.

(Digital, 2017)

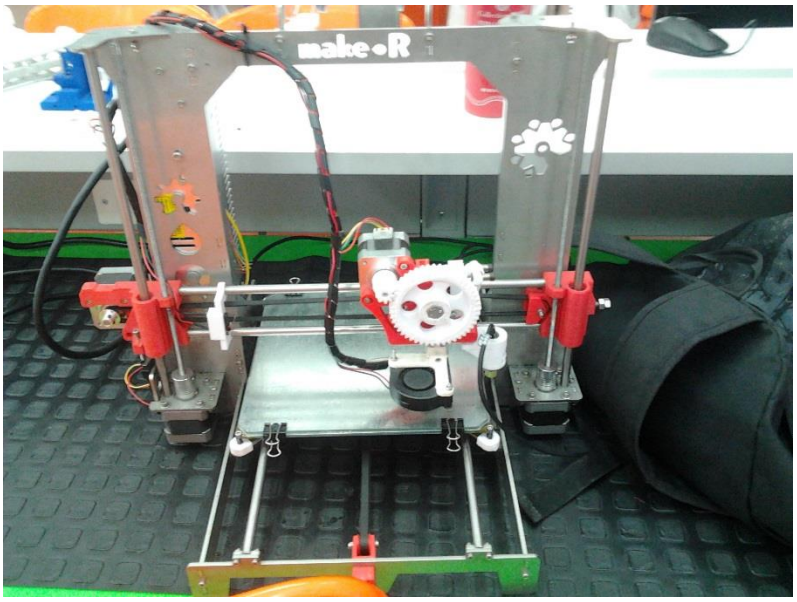

Figura 1. Impresora 3D pascual bravo
Fuente: diseño del realizador proyecto de grado

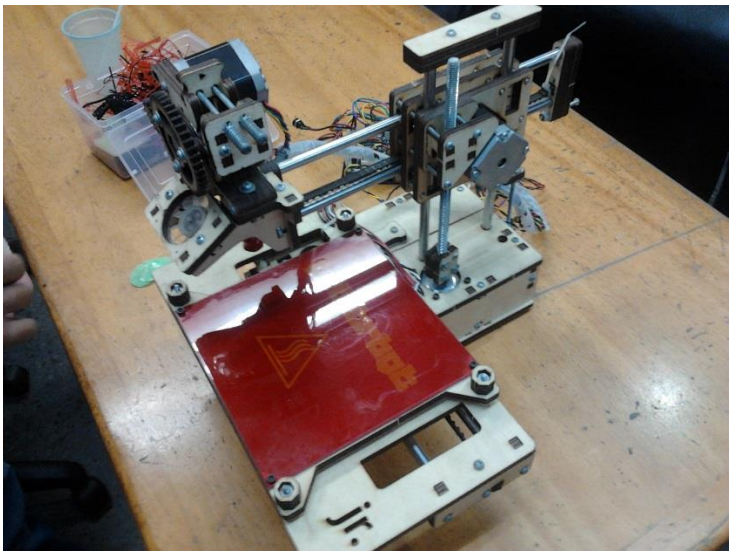

Figura 2. Impresora 3D UdeA
Fuente: diseño del realizador proyecto de grado

Que es arduino?

Arduino es una plataforma electrónica de código abierto basada en hardware y software fáciles de usar. Las placas Arduino son capaces de leer las entradas - la luz en un sensor, un dedo sobre un botón o un mensaje de Twitter - y convertirlo en una salida - la activación de un motor, encender un LED, publicar algo en línea. Usted puede decirle a su junta lo que debe hacer enviando un conjunto de instrucciones al microcontrolador en el tablero. Para ello se utiliza el lenguaje de programación de Arduino (basado en el cableado), y el software de Arduino (IDE), sobre la base de procesamiento.

(salamanca, 2016)

A lo largo de los años Arduino ha sido el cerebro de miles de proyectos, desde objetos cotidianos hasta complejos instrumentos científicos. Una comunidad mundial de los fabricantes - estudiantes, aficionados, artistas, programadores y profesionales - ha reunido en torno a esta plataforma de código abierto, sus contribuciones han añadido hasta una increíble cantidad de conocimiento accesible que puede ser de gran ayuda para los principiantes como para expertos.

(salamanca, 2016)

Arduino nació en el Ivrea Interaction Design Institute como una herramienta fácil para el prototipado rápido, dirigido a estudiantes sin experiencia en electrónica y programación. Tan pronto como llegó a una comunidad más amplia, la placa Arduino comenzó a cambiar para adaptarse a las nuevas necesidades y retos, la diferenciación de su oferta desde simples tablas de 8 bits a los productos de la IO aplicaciones, la impresión 3D portátil y sistemas empotrados. Todas las placas de Arduino son totalmente de código abierto, lo que permite a los usuarios construirlas independientemente y eventualmente adaptarlas a sus necesidades particulares. El software también es de código abierto, y está creciendo a través de las contribuciones de los usuarios en todo el mundo.

(salamanca, 2016)

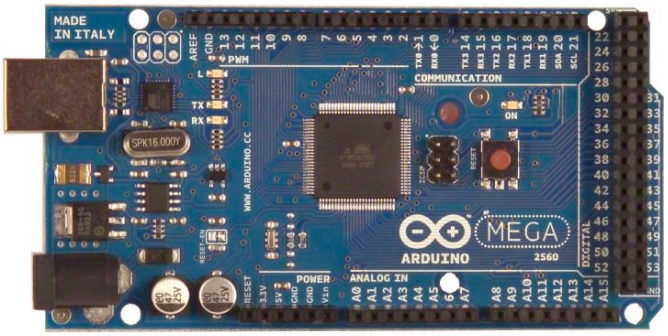

Figura 3. Arduino Mega 2560
Fuente: Tomado de: https://www.adafruit.com/product/50

5. Metodología

5.1 Tipo de Estudio:

El proyecto de grado es de tipo experimental ya que en la impresora se van a utilizar elementos de otros aparatos reciclados que no están diseñados para el uso al que se les va a tratar de dar en la impresora 3D en los sistemas móviles y es posible que no funcione o funcionen y su acople lleve mucho más tiempo.

5.2 Método:

Se va a comenzar con una investigación de las impresoras 3D, que son?, para que sirven?, como funcionan?, para tener una idea de los conceptos básicos de las impresoras 3D y saber cuales son los materiales mas adecuados en la elaboración de esta proyecto; luego se va a buscar y leer un documento donde hable sobre los aparatos eléctricos y electrónicos, que es un documento en pdf llamado Lineamientos Técnicos Para el Manejo de Residuos de Aparatos Eléctricos y Electrónicos (RRAE), Este para entender la problemática global que generan este tipo de aparatos.

En un almacén de electrónica se va a comprar las demás cosas de la impresora que no forman parte de la estructura física que se va a hacer con RAEE como lo son: Driver Pololu, Arduino Mega, Ramps 1.4, Motores Nema 17, Mk8, Extrusor Hotend, Tubo Ptfe, Finales De Carrera, Jumpers, Pla, Conector Ptfe, Tornillos, Arandelas, Tuercas, Soporte Del Extrusor.

Con las 4 impresoras que ya se tienen a la mano se comienzan a desarmar y se les saca a cada una el sistemas de guía donde van los cartuchos de tinta que van a servir como ejes de la impresora (X, Y, Z). las guías de los ejes Y y Z se van a acoplar a una estructura de una carcasa de CPU y el eje X se acopla a los ejes Y para crear los movimientos de los tres ejes, luego se acoplan a cada sistema de movimiento de los ejes los motores paso a

paso bipolares EM 323 de impresoras de papel EPSON, Se coloca el vidrio igualmente reciclado en el eje Z y se coloca a un lado de la estructura de la impresora el sistema mecanico del extrusor que mueve que es el MK8 acoplado al motor NEMA 17 que es el que empuja el PLA hacia el extrusor hotend que se acopla al eje X.

Se colocan los finales de carrera en cada eje para que el sistema sepa cual es el punto cero de cada eje y tenga una referencia del espacio del vidrio donde va a imprimir, se coloca la manija para levar facilmente la impresora de un lugar a otro, luego se coloca la fuente de computador en la parte trasera de la impresora sacando solo los cables amarillo y negro que son 12v y tierra para alimentar la ramps que es el sistema de potencia que alimentara los motores y el extrusor hotend.

Se coloca el arduino en la impresora que es el sistema de control que va a llevar cargado en el un programa que va a controlar todo el sistema, sobre el se colca la ramps y se conectan todos los cables de motores, extrusor hotend, finales de carrera y alimentación.

Se descargan de internet los programas MARLIN y PRONTERFAC: El MARLIN es el programa que se descarga en el arduino y controla la impresora y todas sus variables, y el PRONTERFACE es el programa que se descarga en el computador para darle puesta a punto a los motores y extrusor para que la maquina pueda funcionar correctamente y de ahí tambien se puede imprimir .

Se descarga el MARLIN en el arduino por medio del programa de arduino, y se descarga el PRONTERFACE en el computador y se comienza a darle puesta a punto calibrando drivers de cada motor y los movimientos en centímetro del extrusor hotend en el vidrio y los pasos por centímetro de cada motor para que el trabaje correctamente, se calibra el PID en el MARLIN activando el hotend en el pronterface y el msmo programa al finalizar el calentamiento da unos valores que se deben cambiar en el MARLIN. Luego se descarga un diseño de internet, se monta al PRONTERFACE y la maquina lo comienza a imprimir.

6 Resultados Del Proyecto

Se espera que este proyecto muestre a las personas que se aprende más tratando de construir una impresora de una forma más artesanal solo comprando lo necesario y no comprar todo y solo armarla puesto que así es más sencillo y no se van a presentar muchos inconvenientes que es de lo que se aprende y que con la utilización de accesorios reciclados aparte de aprender más , se está ayudando a minimizar de alguna manera el impacto que generan los residuos de aparatos eléctricos y electrónicos en el medio ambiente, aparte de que es más económica la construcción.

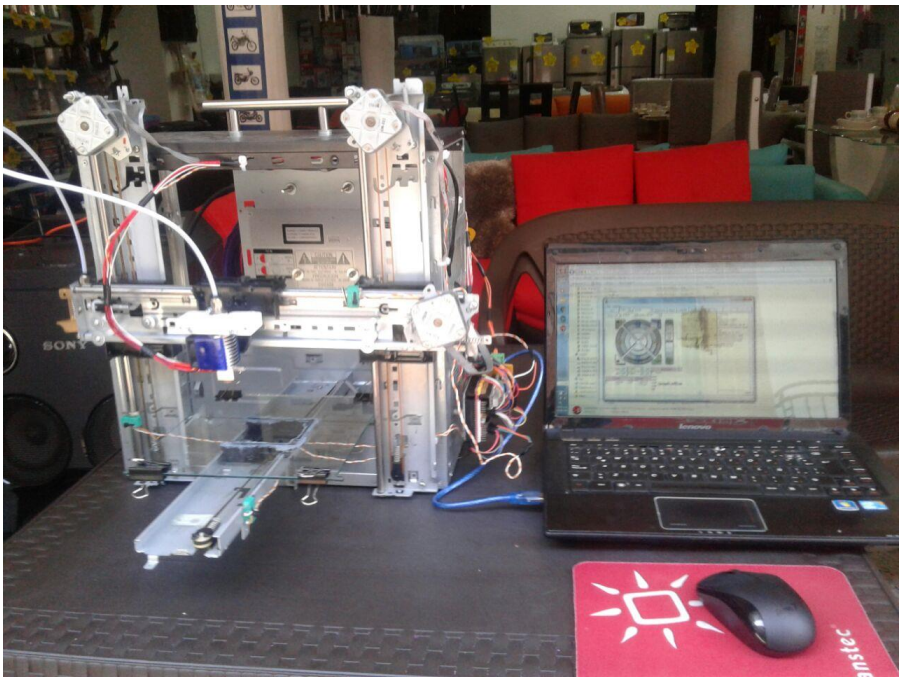

Figura 4. Impresora 3D terminada y pronterface
Fuente: diseño del realizador proyecto de grado

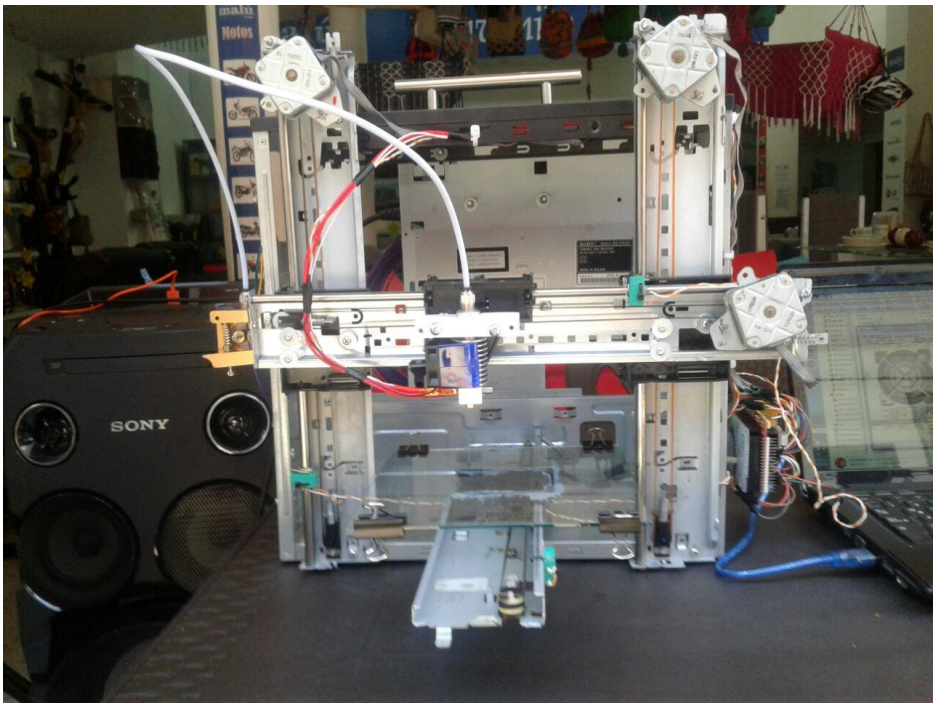

Figura 5. Impresora 3D terminada
Fuente: diseño del realizador proyecto de grado

7. Conclusiones

Es posible la creación de proyectos con la utilización de desechos de aparatos eléctricos y electrónicos reciclados ayudando a incentivar aún más la creatividad.

Se pudo evidenciar que al no conocer proyectos tecnológicos como impresoras 3d, drones, cortadoras cnc, se ven como proyectos muy complicados e inalcanzables de construir, pero al momento de empezar a construirlo y se va conociendo sus partes, cada componente y que hace cada cosa, se van haciendo fáciles de construir.

No se necesita tener muchos recursos económicos al momento de crear un proyecto, solo se necesita creatividad y ganas, ya que con la creatividad se encuentra de alguna u otra manera la forma de darle vida a un proyecto.

Se puede crear ciencia y tecnología de la mano del aprovechamiento de los desperdicios de aparatos tecnológicos ayudando a minimizar los impactos ambientales que estos residuos generan.

Con la construcción de proyectos con residuos de aparatos eléctricos y electrónicos, es un poco complicada la construcción y se necesita de mucha paciencia y normalmente se va a observar mucho desajuste al cual se le debe dedicar más tiempo ya que los residuos que se usan no fueron diseñados para ese fin en específico y por esa razón tendrán desajuste.

8. Recomendaciones

Al momento de realizar algún proyecto con residuos de aparatos electrónicos y electrónicos, debe tener un tiempo prudencial para la construcción, ya que al crear máquinas de esta manera tan artesanal, resultan muchos inconvenientes, puesto que casi nada encaja a la perfección.

Dedicarle un buen tiempo en darle un buen ajuste a todas las partes móviles, ya que como son partes que no son diseñados para esos fines se van a encontrar muchos desajustes.

Investigar con tiempo de cómo sería la construcción con partes originales del proyecto que se desee construir, para luego al momento de utilizar partes recicladas se haga menos complicada el ensamble.

9. Referencias bibliográficas

Digital, M. (4 de 3 de 2017). Obtenido de Mundo Digital Ciencia y Tecnologia: http://www.mundodigital.net/los-beneficios-de-las-impresoras-3d-en-la-ensenanza/

DW. (27 de mayo de 2014). Reciclaje en Togo: una impresora 3D hecha de basura electrónica | Global 3000 recuperado de https://www.youtube.com/watch?v=AyyYxTF7KG8

Fiant. (24 de febrero de 2015). Impresora 3D casera, con partes recicladas, 1ra parte recuperado de https://www.youtube.com/watch?v=tN42gWELsDw

Fiant. (14 de marzo de 2015). Impresora 3D casera, con partes recicladas, 2ra parte

Recuperado de https://www.youtube.com/watch?v=4YRw7a2MZD4

Fiant. (19 de junio de 2015). Impresora 3D casera, con partes recicladas, 3ra parte

Recuperado de https://www.youtube.com/watch?v=ql20oo3024s

manos, T. l. (2016). *TODOMICRO la tecnologia en tus manos.* Obtenido de http://www.todomicro.com.ar/18-arduino

Ministerio del Medio Ambiente, V. y. (2010). *Lineamientos Tecnicos para el Manejo de Residuos de Aparatos Electricos y Electronicos.* Bogota DC Recuperado de http://www.residuoselectronicos.net/wpcontent/uploads/2012/03/Guia_RAEE_MADS_2011-reducida.pdf

Digital, M. (4 de 3 de 2017). Obtenido de Mundo Digital Ciencia y Tecnologia: http://www.mundodigital.net/los-beneficios-de-las-impresoras-3d-en-la-ensenanza/

manos, T. l. (2016). *TODOMICRO la tecnologia en tus manos.* Obtenido de http://www.todomicro.com.ar/18-arduino

Ministerio del Medio Ambiente, V. y. (2010). *Lineamientos Tecnicos para el Manejo de Residuos de Aparatos Electricos y Electronicos.* Bogota DC: http://www.residuoselectronicos.net/wp-content/uploads/2012/03/Guia_RAEE_MADS_2011-reducida.pdf.

salamanca, o. f. (24 de 2 de 2016). *Tips para trabajar en Arduino.* Obtenido de http://tipsparatrabajarenarduino.blogspot.com.co/2016/02/que-es-arduino-arduino-es-una.html

Wikipedia. (5 de 6 de 2017). *Wikipedia*. Obtenido de https://es.wikipedia.org/wiki/Impresora_3D

Wikipedia, c. d. (5 de 6 de 2017). *Wikipedia.* Obtenido de https://es.wikipedia.org/w/index.php?title=Especial:Citar&page=Impresora_3D&id=99637804

10. Anexos

Anexo A. Driver pololu

Figura 6. Driver pololu
Fuente: Tomado de: https://www.electronicaembajadores.com/es/Productos/Detalle/LCMM024/modulos-electronicos/modulos-driver-y-o-controladores-de-motor

Anexo B. Ramps 1.4

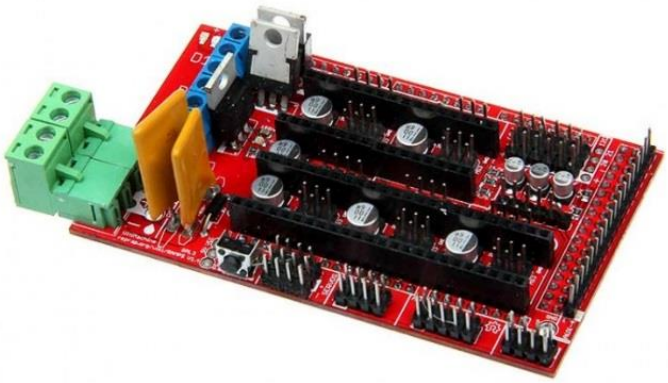

Figura 7. Ramps 1.4
Fuente: Tomado de:http://saber.patagoniatec.com/ramps-impresora-3d-reprap-arduino-argentina-ptec/

Anexo C. Motor nema 17

Figura 8. Motor Nema 17
Fuente: Tomado de: http://tienda.bricogeek.com/motores-paso-a-paso/546-motor-paso-a-paso-nema-17-32kg-cm.html

Anexo D. Motores Epson EM 323

Figura 9. Motores Epson EM 323
Fuente: diseño del realizador proyecto de grado

Anexo E. MK8

Figura 10. MK8
Fuente: Tomado de:
https://www.google.com.co/search?q=mk8+impresoras+3d&source=lnms&tbm=isch&sa=X&ved=0ahUKEwjVooH-85zUAhUD4CYKHZFYAXgQ_AUIBigB&b

Anexo F. Extrusor Hotend

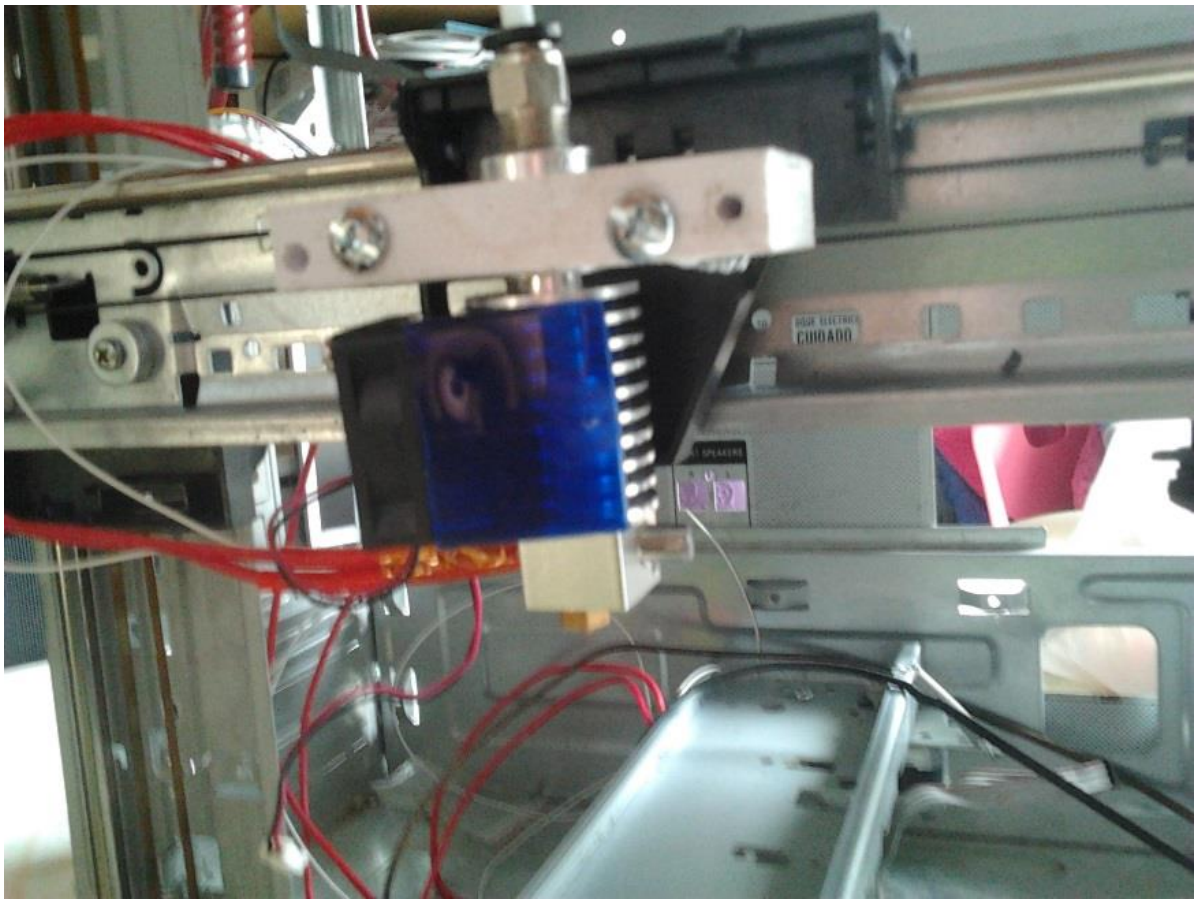

Figura 11. Extrusor hotend
Fuente: diseño del realizador proyecto de grado

Anexo G. Final de carrera

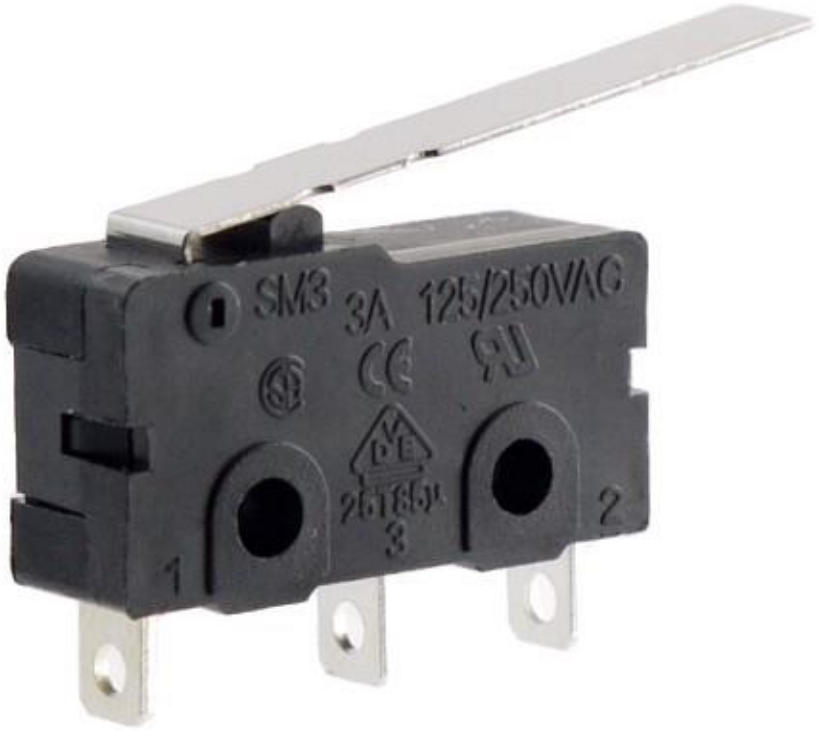

Figura 12. Final de carrera
Fuente: Tomado de:
https://www.google.com.co/search?q=que+es+un+final+de+carrera&source=lnms&tbm=isch&sa=X&ved=0ahUKEwit4J209pzUAhUTxCYKHebsCNwQ_AUI

Anexo H. PLA

Figura 13. PLA
Fuente: Tomado de:
https://www.google.com.co/search?q=que+es+PLA+en+impresión+3d&source=lnms&tbm=isch&sa=X&ved=0ahUKEwjws5Tw-JzUAhVJKiYKHY4VCFoQ_AU

Anexo I. Fuente de computador

Figura 14. Fuente de computador
Fuente: Tomado de:https://www.google.com.co/search?q=fuente+de+computador&tbm=isch&imgil=ATpn8Mb6wpZjpM%253A%253BkU-3pg5v7zrGvM%253Bhttp%25253A%

Anexo J. Tubo PTFE

Figura 15. Tubo PTFE
Fuente: Tomado de:
https://www.google.com.co/search?q=que+es+ptfe&source=lnms&tbm=isch&sa=X&ved=0ahUKEwji-Nik-pzUAhUJSCYKHeTvBhcQ_AUICigB&biw=1304&b

Anexo K. Acoples rapidos

Figura 16. acoples rapidos
Fuente: Tomado de:
https://www.google.com.co/search?q=conectores+rápidos&source=lnms&tbm=isch&sa=X&ved=0ahUKEwiUjeDt_JzUAhUGSSYKHQLEA6kQ_AUIBigB&biw=1304

Anexo L. Impresoras de papel

Figura 17. impresoras de papel
Fuente: diseño del realizador proyecto de grado

Anexo M. Impresora 3D estructura armada

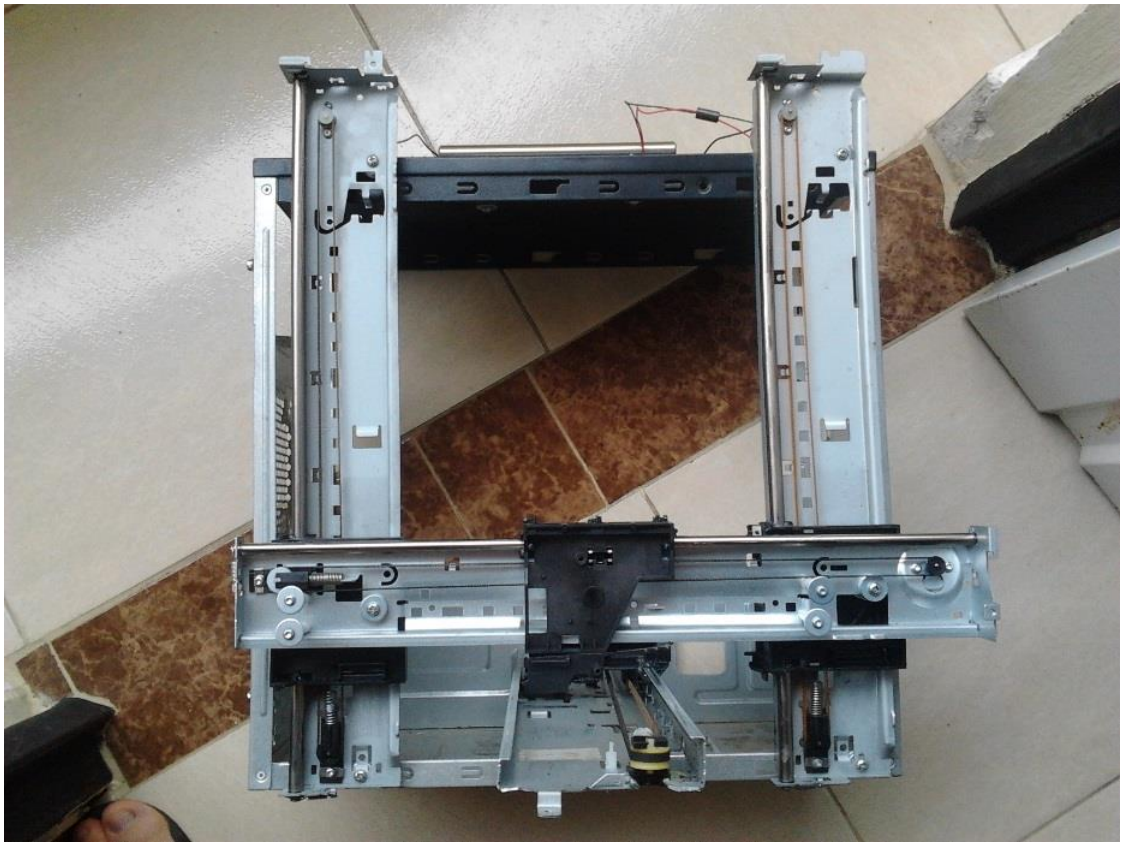

Figura 18. Impresoras 3D estructura armada
Fuente: diseño del realizador proyecto de grado

Anexo N. Acople motores paso a paso epson EM 323 en la impresora

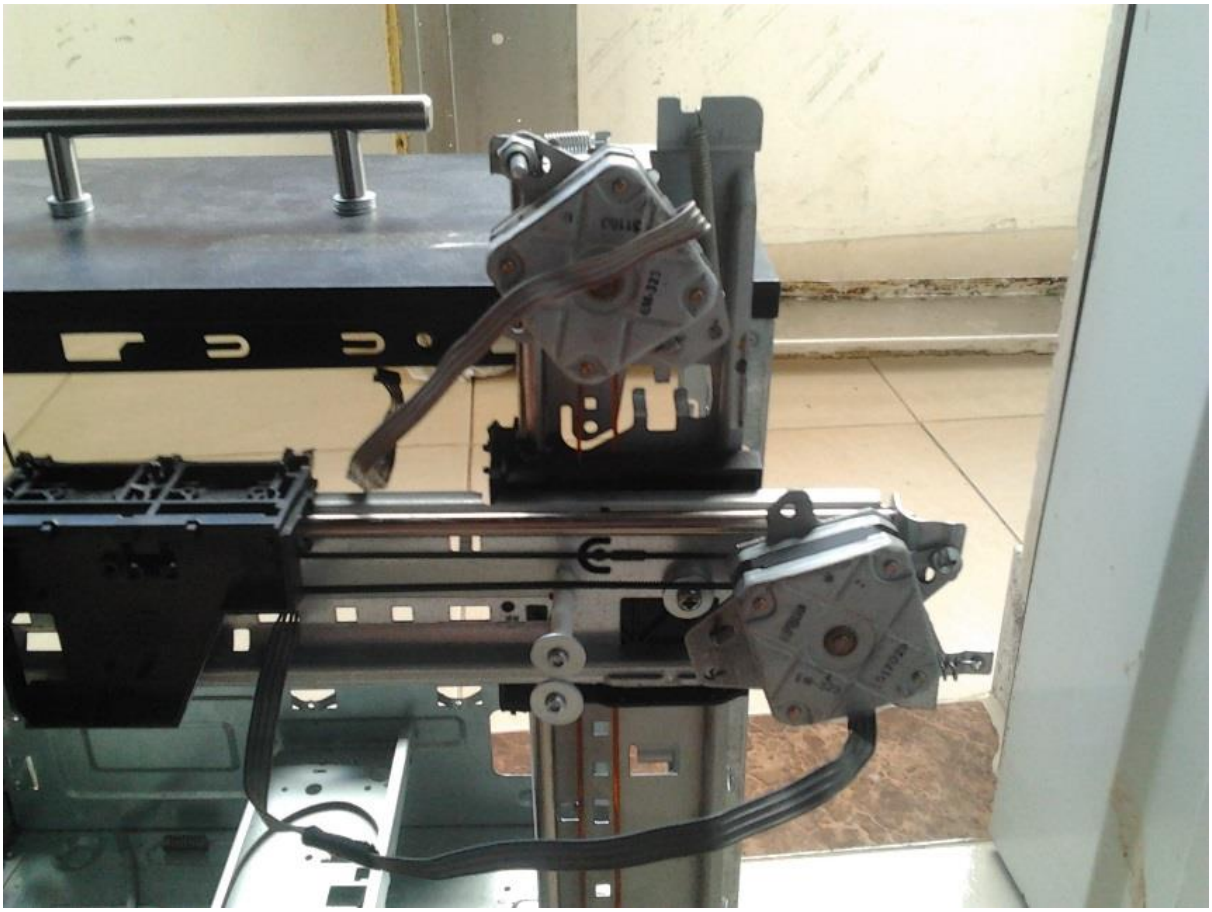

Figura 19. Acople motores paso a paso epson EM 323
Fuente: diseño del realizador proyecto de grado

Anexo O. Sistemas del MK8 acoplado a la impresora

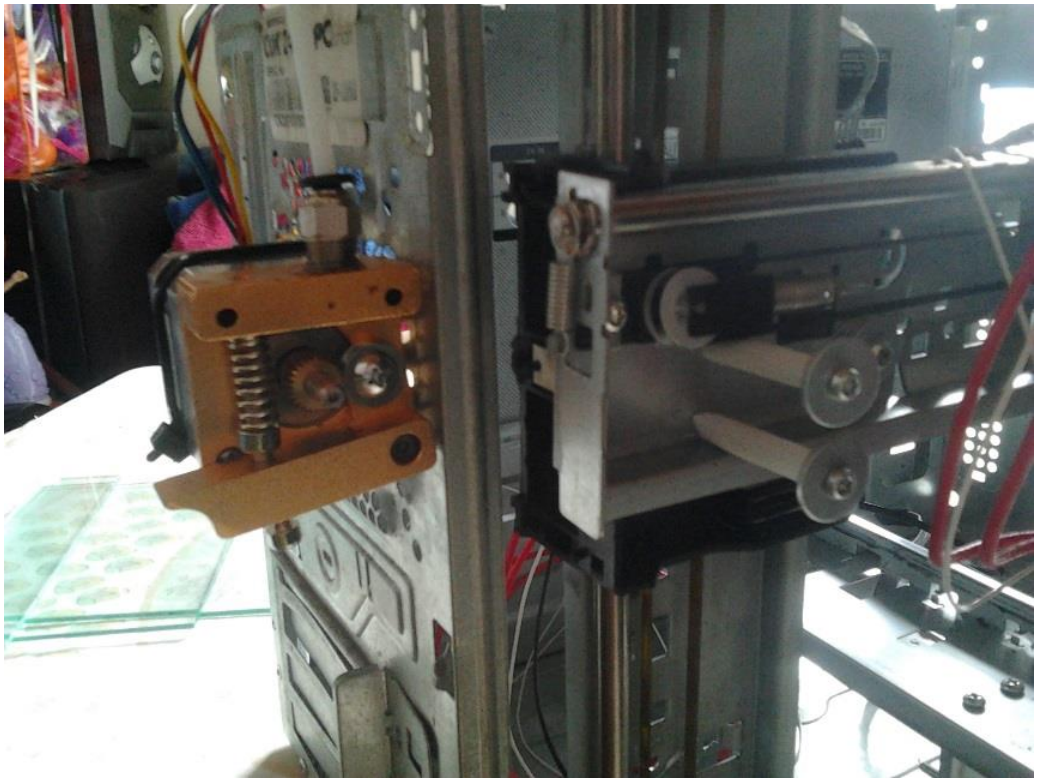

Figura 20. Sistemas del MK8 acoplado a la impresora
Fuente: diseño del realizador proyecto de grado

Anexo P. Extrusor acoplado al eje x

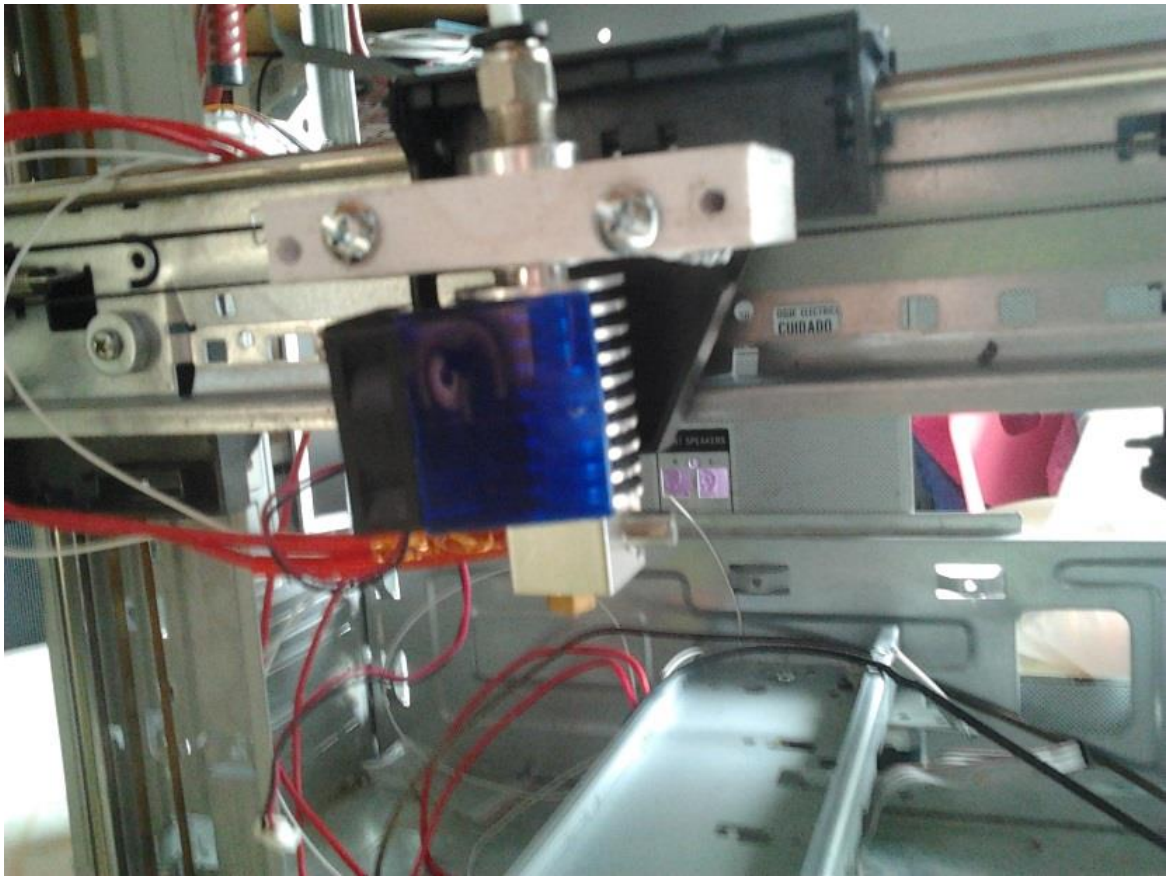

Figura 21. Extrusor acoplado al eje X
Fuente: diseño del realizador proyecto de grado

Anexo Q. Vidrio base puesta en la impresora

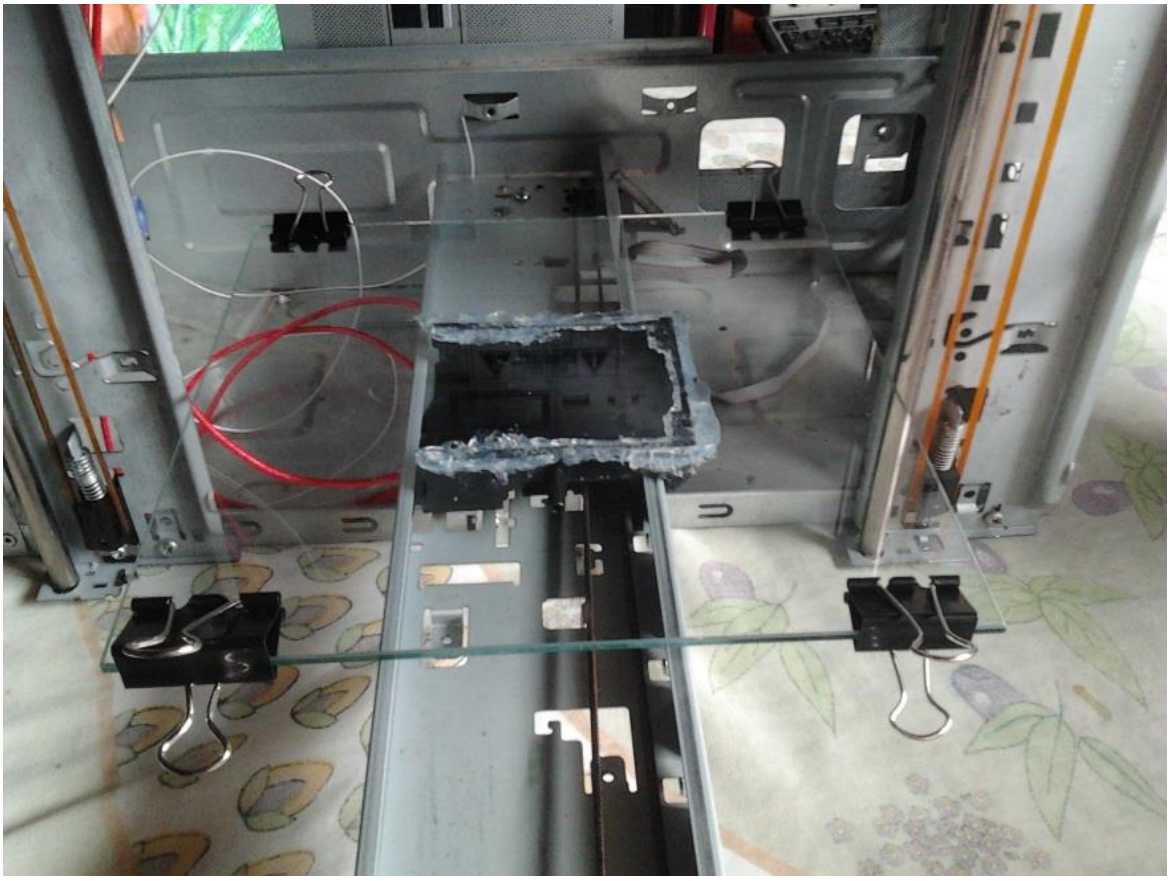

Figura 22. Vidrio base puesta en la impresora
Fuente: diseño del realizador proyecto de grado

Anexo R. Tapa de equipo de sonido

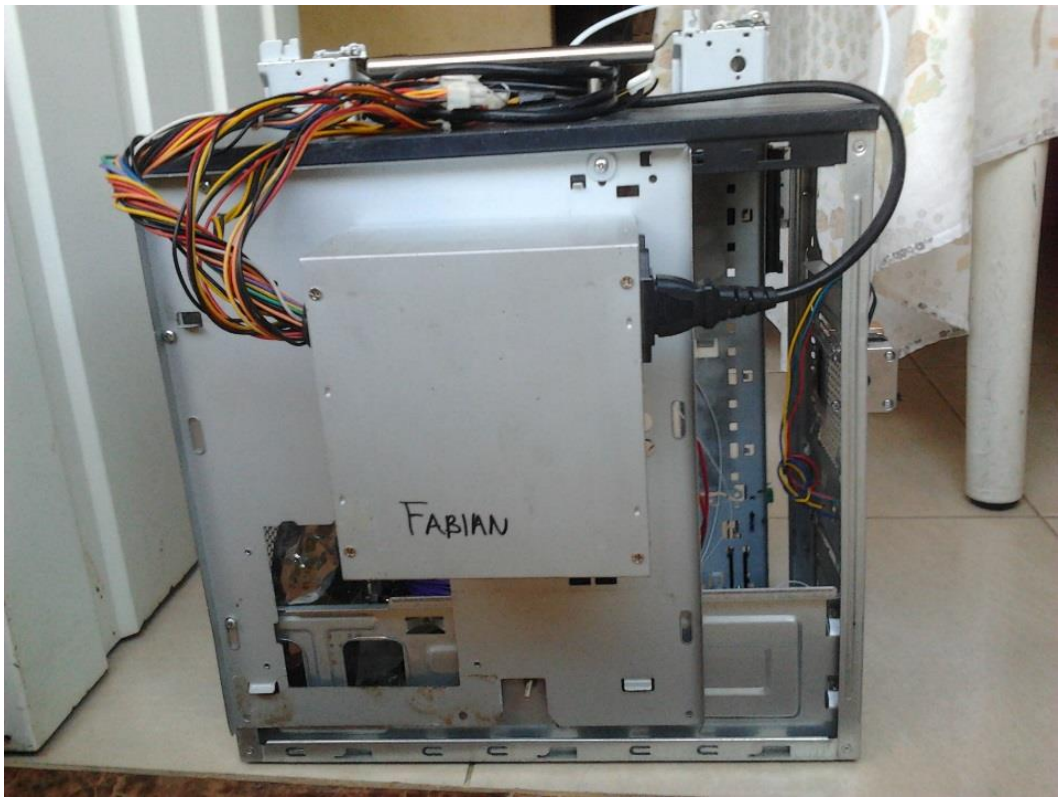

Figura 23. Tapa de equipo de sonido
Fuente: diseño del realizador proyecto de grado

Anexo S. Fuente de alimentación 12v conectada a la ramps

Figura 24. Fuente de alimentación 12v conectada a la ramps
Fuente: diseño del realizador proyecto de grado

Anexo T. Conexiones ramps 1.4

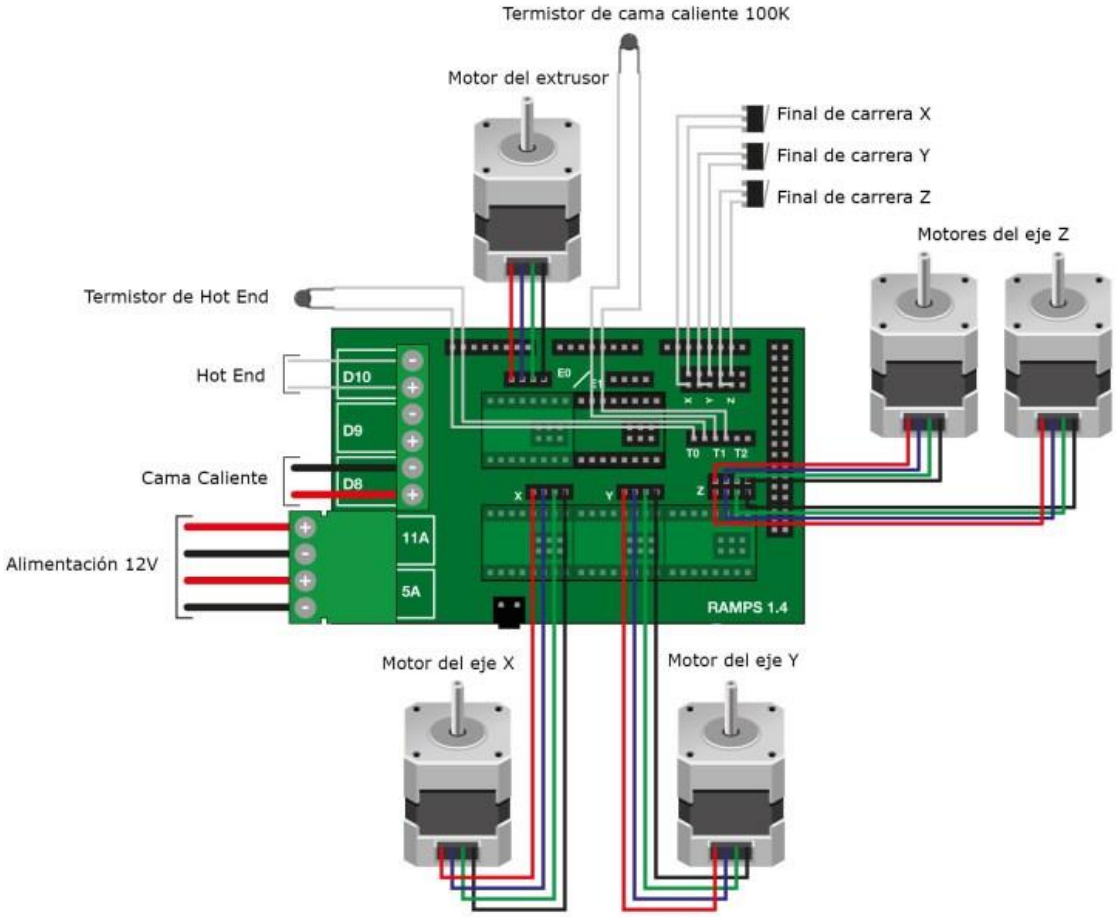

Figura 25. Conexiones ramps 1.4
Fuente: Tomado de: https://electronilab.co/tienda/shield-ramps-1-4-para-impresora-3d-reprap-prusa-mendel/

Anexo U. Imagen marlin

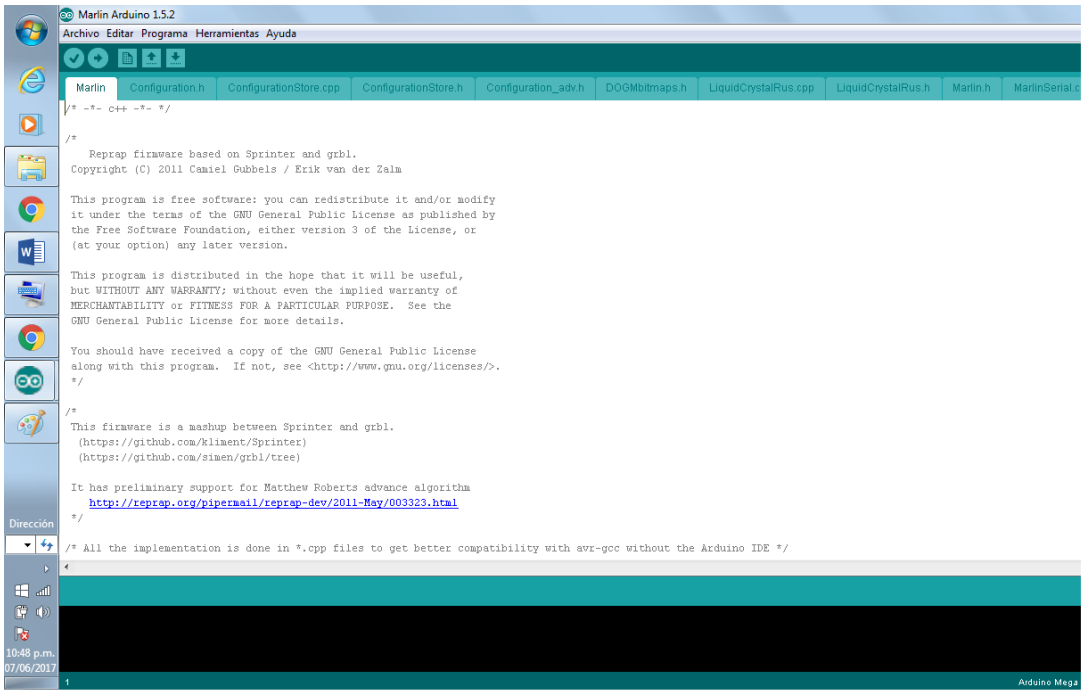

Figura 26. Imagen marlin
Fuente: diseño del realizador proyecto de grado

Anexo V. Impresora 3D armada en su totalidad

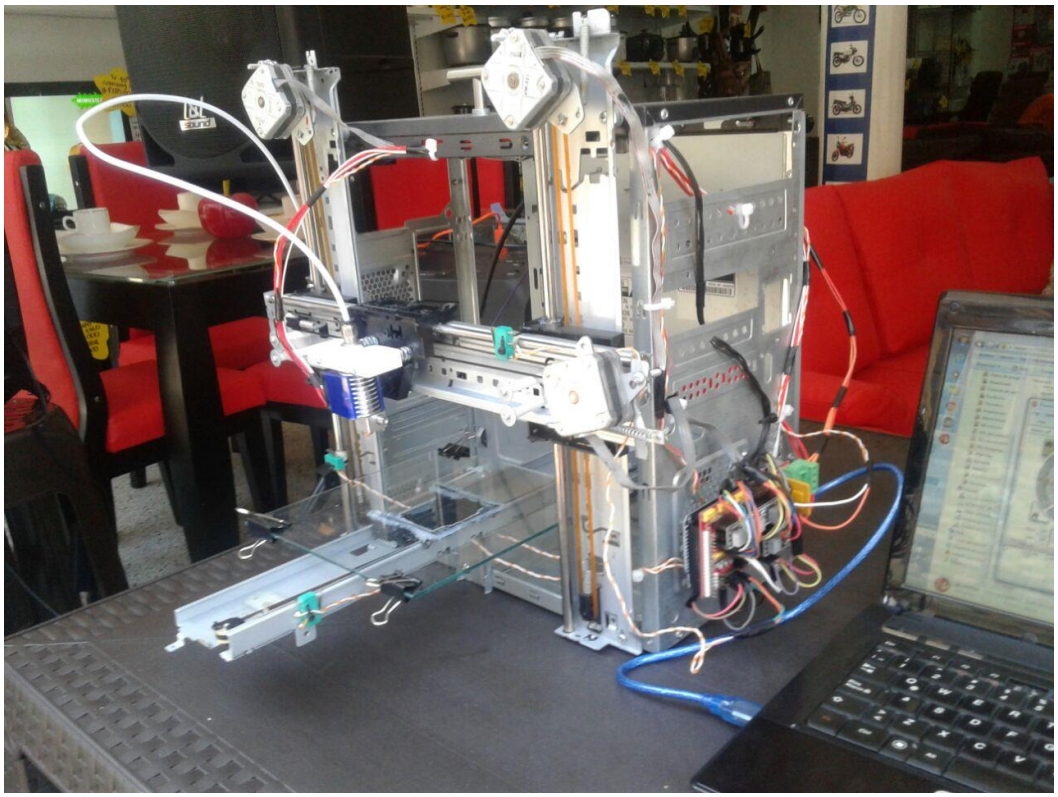

Figura 27. Impresora 3D armada en su totalidad
Fuente: diseño del realizador proyecto de grado

Anexo W. Pronterface

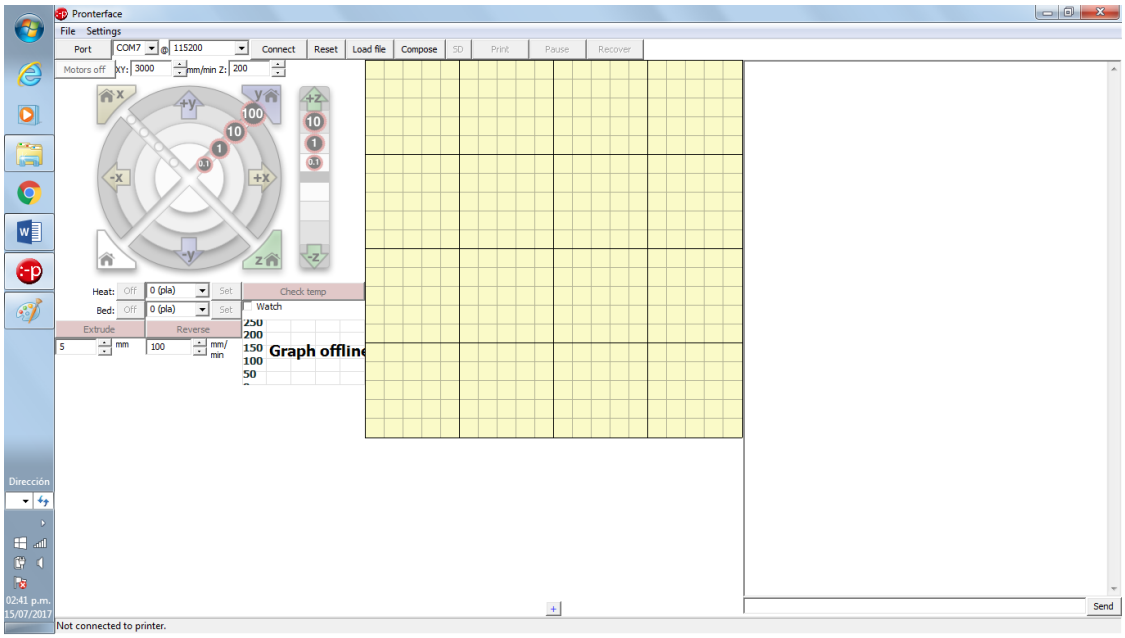

Figura 28. Pronterface
Fuente: diseño del realizador proyecto de grado

Anexo X. Marlin

```
/* -*- c++ -*- */

/*
   Reprap firmware based on Sprinter and grbl.
 Copyright (C) 2011 Camiel Gubbels / Erik van der Zalm

 This program is free software: you can redistribute it and/or modify
 it under the terms of the GNU General Public License as published by
 the Free Software Foundation, either version 3 of the License, or
 (at your option) any later version.

 This program is distributed in the hope that it will be useful,
 but WITHOUT ANY WARRANTY; without even the implied warranty of
 MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 GNU General Public License for more details.

 You should have received a copy of the GNU General Public License
 along with this program. If not, see <http://www.gnu.org/licenses/>.
 */

/*
 This firmware is a mashup between Sprinter and grbl.
  (https://github.com/kliment/Sprinter)
  (https://github.com/simen/grbl/tree)

 It has preliminary support for Matthew Roberts advance algorithm
    http://reprap.org/pipermail/reprap-dev/2011-May/003323.html
```

```
 */

/* All the implementation is done in *.cpp files to get better compatibility with avr-gcc without the Arduino IDE */

/* Use this file to help the Arduino IDE find which Arduino libraries are needed and to keep documentation on GCode */


#include "Configuration.h"

#include "pins.h"


#ifdef ULTRA_LCD

  #if defined(LCD_I2C_TYPE_PCF8575)

    #include <Wire.h>

    #include <LiquidCrystal_I2C.h>

  #elif defined(LCD_I2C_TYPE_MCP23017) || defined(LCD_I2C_TYPE_MCP23008)

    #include <Wire.h>

    #include <LiquidTWI2.h>

  #elif defined(DOGLCD)

    #include <U8glib.h> // library for graphics LCD by Oli Kraus (https://code.google.com/p/u8glib/)

  #else

    #include <LiquidCrystal.h> // library for character LCD

  #endif

#endif


#if defined(DIGIPOTSS_PIN) && DIGIPOTSS_PIN > -1

#include <SPI.h>

#endif
```

```
#ifndef CONFIGURATION_H
#define CONFIGURATION_H

// This configurtion file contains the basic settings.
// Advanced settings can be found in Configuration_adv.h
// BASIC SETTINGS: select your board type, temperature sensor type, axis scaling, and endstop configuration

//User specified version info of this build to display in [Pronterface, etc] terminal window during startup.
//Implementation of an idea by Prof Braino to inform user that any changes made
//to this build by the user have been successfully uploaded into firmware.
#define STRING_VERSION_CONFIG_H __DATE__ " " __TIME__ // build date and time
#define STRING_CONFIG_H_AUTHOR "Darth Raul" //Who made the changes.

// SERIAL_PORT selects which serial port should be used for communication with the host.
// This allows the connection of wireless adapters (for instance) to non-default port pins.
// Serial port 0 is still used by the Arduino bootloader regardless of this setting.
#define SERIAL_PORT 0

// This determines the communication speed of the printer
#define BAUDRATE 115200
//#define BAUDRATE 115200

//// The following define selects which electronics board you have. Please choose the one that matches your setup
// 10 = Gen7 custom (Alfons3 Version) "https://github.com/Alfons3/Generation_7_Electronics"
// 11 = Gen7 v1.1, v1.2 = 11
```

```
// 12 = Gen7 v1.3
// 13 = Gen7 v1.4
// 3  = MEGA/RAMPS up to 1.2 = 3
// 33 = RAMPS 1.3 / 1.4 (Power outputs: Extruder, Bed, Fan)
// 34 = RAMPS 1.3 / 1.4 (Power outputs: Extruder0, Extruder1, Bed)
// 4  = Duemilanove w/ ATMega328P pin assignment
// 5  = Gen6
// 51 = Gen6 deluxe
// 6  = Sanguinololu < 1.2
// 62 = Sanguinololu 1.2 and above
// 63 = Melzi
// 64 = STB V1.1
// 7  = Ultimaker
// 71 = Ultimaker (Older electronics. Pre 1.5.4. This is rare)
// 8  = Teensylu
// 80 = Rumba
// 81 = Printrboard (AT90USB1286)
// 82 = Brainwave (AT90USB646)
// 9  = Gen3+
// 70 = Megatronics
// 701= Megatronics v2.0
// 702= Minitronics v1.0
// 90 = Alpha OMCA board
// 91 = Final OMCA board
// 301 = Rambo

#ifndef MOTHERBOARD
#define MOTHERBOARD 33
```

```
#endif

// This defines the number of extruders
#define EXTRUDERS 1

//// The following define selects which power supply you have. Please choose the one that matches your setup
// 1 = ATX
// 2 = X-Box 360 203Watts (the blue wire connected to PS_ON and the red wire to VCC)

#define POWER_SUPPLY 1

//===========================================================================
//=============================Thermal Settings============================
//===========================================================================
//
//--NORMAL IS 4.7kohm PULLUP!-- 1kohm pullup can be used on hotend sensor, using correct resistor and table
//
//// Temperature sensor settings:
// -2 is thermocouple with MAX6675 (only for sensor 0)
// -1 is thermocouple with AD595
// 0 is not used
// 1 is 100k thermistor - best choice for EPCOS 100k (4.7k pullup)
// 2 is 200k thermistor - ATC Semitec 204GT-2 (4.7k pullup)
// 3 is mendel-parts thermistor (4.7k pullup)
// 4 is 10k thermistor !! do not use it for a hotend. It gives bad resolution at high temp. !!
```

```
// 5 is 100K thermistor - ATC Semitec 104GT-2 (Used in ParCan) (4.7k pullup)

// 6 is 100k EPCOS - Not as accurate as table 1 (created using a fluke thermocouple) (4.7k pullup)

// 7 is 100k Honeywell thermistor 135-104LAG-J01 (4.7k pullup)

// 8 is 100k 0603 SMD Vishay NTCS0603E3104FXT (4.7k pullup)

// 9 is 100k GE Sensing AL03006-58.2K-97-G1 (4.7k pullup)

// 10 is 100k RS thermistor 198-961 (4.7k pullup)

//

//  1k ohm pullup tables - This is not normal, you would have to have changed out your 4.7k for 1k

//                      (but gives greater accuracy and more stable PID)

// 51 is 100k thermistor - EPCOS (1k pullup)

// 52 is 200k thermistor - ATC Semitec 204GT-2 (1k pullup)

// 55 is 100k thermistor - ATC Semitec 104GT-2 (Used in ParCan) (1k pullup)


#define TEMP_SENSOR_0 1

#define TEMP_SENSOR_1 0

#define TEMP_SENSOR_2 0

#define TEMP_SENSOR_BED 0


// Actual temperature must be close to target for this long before M109 returns success

#define TEMP_RESIDENCY_TIME 10 // (seconds)

#define TEMP_HYSTERESIS 3      // (degC) range of +/- temperatures considered "close" to the target one

#define TEMP_WINDOW     1      // (degC) Window around target to start the recidency timer x degC early.


// The minimal temperature defines the temperature below which the heater will not be enabled It is used

// to check that the wiring to the thermistor is not broken.
```

```
// Otherwise this would lead to the heater being powered on all the time.
#define HEATER_0_MINTEMP 5
#define HEATER_1_MINTEMP 5
#define HEATER_2_MINTEMP 5
#define BED_MINTEMP 5

// When temperature exceeds max temp, your heater will be switched off.
// This feature exists to protect your hotend from overheating accidentally, but *NOT* from thermistor short/failure!
// You should use MINTEMP for thermistor short/failure protection.
#define HEATER_0_MAXTEMP 260
#define HEATER_1_MAXTEMP 275
#define HEATER_2_MAXTEMP 275
#define BED_MAXTEMP 130

// If your bed has low resistance e.g. .6 ohm and throws the fuse you can duty cycle it to reduce the
// average current. The value should be an integer and the heat bed will be turned on for 1 interval of
// HEATER_BED_DUTY_CYCLE_DIVIDER intervals.
//#define HEATER_BED_DUTY_CYCLE_DIVIDER 4

// PID settings:
// Comment the following line to disable PID and enable bang-bang.
#define PIDTEMP
#define BANG_MAX 256 // limits current to nozzle while in bang-bang mode; 256=full current
#define PID_MAX 256 // limits current to nozzle while PID is active (see PID_FUNCTIONAL_RANGE below); 256=full current
#ifdef PIDTEMP
```

```
  //#define  PID_DEBUG  // Sends debug data to the serial port.

  //#define  PID_OPENLOOP 1 // Puts PID in open loop. M104/M140 sets the output power
from 0 to PID_MAX

  #define  PID_FUNCTIONAL_RANGE  10 // If the temperature  difference  between the
target temperature  and the actual temperature

                                   // is more then PID_FUNCTIONAL_RANGE  then the PID will  be
shut  off and the heater will  be set to min/max.  (10)

  #define  PID_INTEGRAL_DRIVE_MAX  255  //limit  for the integral  term

  #define  K1 0.95 //smoothing  factor withing  the PID

  #define  PID_dT ((16.0 * 8.0)/(F_CPU / 64.0 / 256.0)) //sampling  period of the
temperature  routine


// If you are using  a preconfigured  hotend  then you can use one of the value  sets by
uncommenting  it

// Ultimaker

    #define  DEFAULT_Kp  20.29

    #define  DEFAULT_Ki  1.39

    #define  DEFAULT_Kd  73.80


// Makergear

//    #define  DEFAULT_Kp  7.0

//    #define  DEFAULT_Ki  0.1

//    #define  DEFAULT_Kd  12


// Mendel Parts V9 on 12V

//    #define  DEFAULT_Kp  63.0

//    #define  DEFAULT_Ki  2.25

//    #define  DEFAULT_Kd  440

#endif  // PIDTEMP
```

```
// Bed Temperature Control

// Select PID or bang-bang with PIDTEMPBED. If bang-bang,
BED_LIMIT_SWITCHING will enable hysteresis

//

// uncomment this to enable PID on the bed. It uses the same frequency PWM as the
extruder.

// If your PID_dT above is the default, and correct for your hardware/configuration, that
means 7.689Hz,

// which is fine for driving a square wave into a resistive load and does not significantly
impact you FET heating.

// This also works fine on a Fotek SSR-10DA Solid State Relay into a 250W heater.

// If your configuration is significantly different than this and you don't understand the
issues involved, you proabaly

// shouldn't use bed PID until someone else verifies your hardware works.

// If this is enabled, find your own PID constants below.

//#define PIDTEMPBED

//

//#define BED_LIMIT_SWITCHING


// This sets the max power delived to the bed, and replaces the
HEATER_BED_DUTY_CYCLE_DIVIDER option.

// all forms of bed control obey this (PID, bang-bang, bang-bang with hysteresis)

// setting this to anything other than 256 enables a form of PWM to the bed just like
HEATER_BED_DUTY_CYCLE_DIVIDER did,

// so you shouldn't use it unless you are OK with PWM on your bed. (see the comment on
enabling PIDTEMPBED)

#define MAX_BED_POWER 256 // limits duty cycle to bed; 256=full current


#ifdef PIDTEMPBED

//120v 250W silicone heater into 4mm borosilicate (MendelMax 1.5+)
```

```
//from FOPDT model - kp=.39 Tp=405 Tdead=66, Tc set to 79.2, argressive factor of .15 (vs .1, 1, 10)
  #define DEFAULT_bedKp 10.00
  #define DEFAULT_bedKi .023
  #define DEFAULT_bedKd 305.4

//120v 250W silicone heater into 4mm borosilicate (MendelMax 1.5+)
//from pidautotune
//  #define DEFAULT_bedKp 97.1
//  #define DEFAULT_bedKi 1.41
//  #define DEFAULT_bedKd 1675.16

// FIND YOUR OWN: "M303 E-1 C8 S90" to run autotune on the bed at 90 degreesC for 8 cycles.
#endif // PIDTEMPBED


//this prevents dangerous Extruder moves, i.e. if the temperature is under the limit
//can be software-disabled for whatever purposes by
#define PREVENT_DANGEROUS_EXTRUDE
//if PREVENT_DANGEROUS_EXTRUDE is on, you can still disable (uncomment) very long bits of extrusion separately.
#define PREVENT_LENGTHY_EXTRUDE

#define EXTRUDE_MINTEMP 170
#define EXTRUDE_MAXLENGTH (X_MAX_LENGTH+Y_MAX_LENGTH) //prevent extrusion of very large distances.
```

```
//============================================================================
==========

//============================Mechanical
Settings============================

//============================================================================
==========


// Uncomment the following line to enable CoreXY kinematics

// #define COREXY


// corse Endstop Settings

#define ENDSTOPPULLUPS // Comment this out (using // at the start of the line) to
disable the endstop pullup resistors


#ifndef ENDSTOPPULLUPS

 // fine Enstop settings: Individual Pullups. will be ignord if ENDSTOPPULLUPS is
defined

 #define ENDSTOPPULLUP_XMAX

 #define ENDSTOPPULLUP_YMAX

 #define ENDSTOPPULLUP_ZMAX

 #define ENDSTOPPULLUP_XMIN

 #define ENDSTOPPULLUP_YMIN

 //#define ENDSTOPPULLUP_ZMIN

#endif


#ifdef ENDSTOPPULLUPS

 #define ENDSTOPPULLUP_XMAX

 #define ENDSTOPPULLUP_YMAX

 #define ENDSTOPPULLUP_ZMAX

 #define ENDSTOPPULLUP_XMIN
```

```
  #define ENDSTOPPULLUP_YMIN
  #define ENDSTOPPULLUP_ZMIN
#endif

// The pullups are needed if you directly connect a mechanical endswitch between the signal and ground pins.
const bool X_ENDSTOPS_INVERTING = true; // set to true to invert the logic of the endstops.
const bool Y_ENDSTOPS_INVERTING = true; // set to true to invert the logic of the endstops.
const bool Z_ENDSTOPS_INVERTING = true; // set to true to invert the logic of the endstops.
//#define DISABLE_MAX_ENDSTOPS

// For Inverting Stepper Enable Pins (Active Low) use 0, Non Inverting (Active High) use 1
#define X_ENABLE_ON 0
#define Y_ENABLE_ON 0
#define Z_ENABLE_ON 0
#define E_ENABLE_ON 0 // For all extruders

// Disables axis when it's not being used.
#define DISABLE_X false
#define DISABLE_Y false
#define DISABLE_Z true
#define DISABLE_E false // For all extruders

#define INVERT_X_DIR false    // for Mendel set to false, for Orca set to true
#define INVERT_Y_DIR false    // for Mendel set to true, for Orca set to false
#define INVERT_Z_DIR true     // for Mendel set to false, for Orca set to true
```

```
#define INVERT_E0_DIR false   // for direct drive extruder v9 set to true, for geared extruder set to false

#define INVERT_E1_DIR false   // for direct drive extruder v9 set to true, for geared extruder set to false

#define INVERT_E2_DIR false   // for direct drive extruder v9 set to true, for geared extruder set to false


// ENDSTOP SETTINGS:

// Sets direction of endstops when homing; 1=MAX, -1=MIN

#define X_HOME_DIR -1

#define Y_HOME_DIR -1

#define Z_HOME_DIR -1


#define min_software_endstops true //If true, axis won't move to coordinates less than HOME_POS.

#define max_software_endstops true  //If true, axis won't move to coordinates greater than the defined lengths below.

// Travel limits after homing

#define X_MAX_POS 100

#define X_MIN_POS 0

#define Y_MAX_POS 165

#define Y_MIN_POS 0

#define Z_MAX_POS 150

#define Z_MIN_POS 0


#define X_MAX_LENGTH (X_MAX_POS - X_MIN_POS)

#define Y_MAX_LENGTH (Y_MAX_POS - Y_MIN_POS)

#define Z_MAX_LENGTH (Z_MAX_POS - Z_MIN_POS)


// The position of the homing switches
```

```
//#define MANUAL_HOME_POSITIONS  // If defined, MANUAL_*_HOME_POS
below will be used

//#define BED_CENTER_AT_0_0  // If defined, the center of the bed is at (X=0, Y=0)


//Manual homing switch locations:

#define MANUAL_X_HOME_POS 0

#define MANUAL_Y_HOME_POS 0

#define MANUAL_Z_HOME_POS 0


//// MOVEMENT SETTINGS

#define NUM_AXIS 4 // The axis order in all axis related arrays is X, Y, Z, E

#define HOMING_FEEDRATE {2000, 2000, 100, 0} // set the homing speeds (mm/min)
[50*60,50*60, 4*60, 0]


// default settings


#define DEFAULT_AXIS_STEPS_PER_UNIT  {113.311,266.6,130.5,191.2} // default
steps per unit for ultimaker

#define DEFAULT_MAX_FEEDRATE          {330, 330, 3.3, 45}    // (mm/sec)

#define DEFAULT_MAX_ACCELERATION      {1500,1500,100,10000}    // X, Y, Z, E
maximum start speed for accelerated moves. E default values are good for skeinforge 40+,
for older versions raise them a lot.

                                      //(9000,9000,100,10000)

#define DEFAULT_ACCELERATION          1000    // X, Y, Z and E max acceleration in
mm/s^2 for printing moves [3000]

#define DEFAULT_RETRACT_ACCELERATION  2000   // X, Y, Z and E max
acceleration in mm/s^2 for r retracts [3000]


// Offset of the extruders (uncomment if using more than one and relying on firmware to
position when changing).

// The offset has to be X=0, Y=0 for the extruder 0 hotend (default extruder).
```

// For the other hotends it is their distance from the extruder 0 hotend.

// #define EXTRUDER_OFFSET_X {0.0, 20.00} // (in mm) for each extruder, offset of the hotend on the X axis

// #define EXTRUDER_OFFSET_Y {0.0, 5.00} // (in mm) for each extruder, offset of the hotend on the Y axis

// The speed change that does not require acceleration (i.e. the software might assume it can be done instanteneously)

#define DEFAULT_XYJERK 15 // (mm/sec) [20.0]

#define DEFAULT_ZJERK 0.4 // (mm/sec)

#define DEFAULT_EJERK 5 // (mm/sec) [5.0]

//===

//=============================Additional Features=============================

//===

// EEPROM

// the microcontroller can store settings in the EEPROM, e.g. max velocity...

// M500 - stores paramters in EEPROM

// M501 - reads parameters from EEPROM (if you need reset them after you changed them temporarily).

// M502 - reverts to the default "factory settings". You still need to store them in EEPROM afterwards if you want to.

//define this to enable eeprom support

//#define EEPROM_SETTINGS

//to disable EEPROM Serial responses and decrease program space by ~1700 byte: comment this out:

// please keep turned on if you can.

```
//#define EEPROM_CHITCHAT

// Preheat Constants
#define PLA_PREHEAT_HOTEND_TEMP 200
#define PLA_PREHEAT_HPB_TEMP 50
#define PLA_PREHEAT_FAN_SPEED 255 // Insert Value between 0 and 255

#define ABS_PREHEAT_HOTEND_TEMP 240
#define ABS_PREHEAT_HPB_TEMP 100
#define ABS_PREHEAT_FAN_SPEED 200 // Insert Value between 0 and 255 (tenia 255)

//LCD and SD support
//#define ULTRA_LCD //general lcd support, also 16x2
//#define DOGLCD // Support for SPI LCD 128x64 (Controller ST7565R graphic Display Family)
//#define SDSUPPORT // Enable SD Card Support in Hardware Console
//#define SDSLOW // Use slower SD transfer mode (not normally needed - uncomment if you're getting volume init error)

//#define ULTIMAKERCONTROLLER //as available from the ultimaker online store.
//#define ULTIPANEL //the ultipanel as on thingiverse

// The RepRapDiscount Smart Controller (white PCB)
// http://reprap.org/wiki/RepRapDiscount_Smart_Controller
#define REPRAP_DISCOUNT_SMART_CONTROLLER

// The GADGETS3D G3D LCD/SD Controller (blue PCB)
// http://reprap.org/wiki/RAMPS_1.3/1.4_GADGETS3D_Shield_with_Panel
```

```
//#define  G3D_PANEL


//The RepRapDiscount FULL GRAPHIC Smart Controller (quadratic white PCB)
// http://reprap.org/wiki/RepRapDiscount_Full_Graphic_Smart_Controller
//
// ==> REMEMBER TO INSTALL U8glib to your ARDUINO library folder:
http://code.google.com/p/u8glib/wiki/u8glib
//#define  REPRAP_DISCOUNT_FULL_GRAPHIC_SMART_CONTROLLER


// The RepRapWorld REPRAPWORLD_KEYPAD v1.1
// http://reprapworld.com/?products_details&products_id=202&cPath=1591_1626
//#define  REPRAPWORLD_KEYPAD
//#define  REPRAPWORLD_KEYPAD_MOVE_STEP  10.0 // how much should be moved
when a key is pressed, eg 10.0 means 10mm per click


//automatic expansion
#if defined (REPRAP_DISCOUNT_FULL_GRAPHIC_SMART_CONTROLLER)
 #define  DOGLCD
 #define  U8GLIB_ST7920
 #define  REPRAP_DISCOUNT_SMART_CONTROLLER
#endif


#if defined(ULTIMAKERCONTROLLER) ||
defined(REPRAP_DISCOUNT_SMART_CONTROLLER) || defined(G3D_PANEL)
 #define  ULTIPANEL
 #define  NEWPANEL
#endif


#if defined(REPRAPWORLD_KEYPAD)
```

```
  #define  NEWPANEL
  #define  ULTIPANEL
#endif

//I2C PANELS

//#define  LCD_I2C_SAINSMART_YWROBOT
#ifdef  LCD_I2C_SAINSMART_YWROBOT
  // This uses the LiquidCrystal_I2C  library  ( https://bitbucket.org/fmalpartida/new-liquidcrystal/wiki/Home  )
  // Make sure it is placed in the Arduino  libraries  directory.
  #define  LCD_I2C_TYPE_PCF8575
  #define  LCD_I2C_ADDRESS  0x27   // I2C Address of the port expander
  #define  NEWPANEL
  #define  ULTIPANEL
#endif

// PANELOLU2 LCD with  status LEDs,  separate encoder and click  inputs
//#define  LCD_I2C_PANELOLU2
#ifdef  LCD_I2C_PANELOLU2
  // This uses the LiquidTWI2  library  v1.2.3 or later ( https://github.com/lincomatic/LiquidTWI2  )
  // Make sure the LiquidTWI2  directory  is placed in the Arduino  or Sketchbook libraries subdirectory.
  // (v1.2.3 no longer  requires  you to define  PANELOLU  in the LiquidTWI2.h  library  header file)
  // Note: The PANELOLU2  encoder click  input  can either be directly  connected  to a pin
  //       (if BTN_ENC defined  to != -1) or read through  I2C (when  BTN_ENC == -1).
  #define  LCD_I2C_TYPE_MCP23017
  #define  LCD_I2C_ADDRESS  0x20 // I2C Address of the port expander
```

```
  #define LCD_USE_I2C_BUZZER //comment out to disable buzzer on LCD
  #define NEWPANEL
  #define ULTIPANEL
#endif

// Panucatt VIKI LCD with status LEDs, integrated click & L/R/U/P buttons, separate encoder inputs
//#define LCD_I2C_VIKI
#ifdef LCD_I2C_VIKI
  // This uses the LiquidTWI2 library v1.2.3 or later ( https://github.com/lincomatic/LiquidTWI2 )
  // Make sure the LiquidTWI2 directory is placed in the Arduino or Sketchbook libraries subdirectory.
  // Note: The pause/stop/resume LCD button pin should be connected to the Arduino
  //       BTN_ENC pin (or set BTN_ENC to -1 if not used)
  #define LCD_I2C_TYPE_MCP23017
  #define LCD_I2C_ADDRESS 0x20 // I2C Address of the port expander
  #define LCD_USE_I2C_BUZZER //comment out to disable buzzer on LCD (requires LiquidTWI2 v1.2.3 or later)
  #define NEWPANEL
  #define ULTIPANEL
#endif

#ifdef ULTIPANEL
// #define NEWPANEL //enable this if you have a click-encoder panel
  #define SDSUPPORT
  #define ULTRA_LCD
  #ifdef DOGLCD // Change number of lines to match the DOG graphic display
    #define LCD_WIDTH 20
    #define LCD_HEIGHT 5
```

```
  #else
    #define LCD_WIDTH 20
    #define LCD_HEIGHT 4
  #endif
#else //no panel but just lcd
  #ifdef ULTRA_LCD
  #ifdef DOGLCD // Change number of lines to match the 128x64 graphics display
    #define LCD_WIDTH 20
    #define LCD_HEIGHT 5
  #else
    #define LCD_WIDTH 16
    #define LCD_HEIGHT 2
  #endif
  #endif
#endif

// Increase the FAN pwm frequency. Removes the PWM noise but increases heating in the FET/Arduino
//#define FAST_PWM_FAN

// M240 Triggers a camera by emulating a Canon RC-1 Remote
// Data from: http://www.doc-diy.net/photo/rc-1_hacked/
// #define PHOTOGRAPH_PIN     23

// SF send wrong arc g-codes when using Arc Point as fillet procedure
//#define SF_ARC_FIX

// Support for the BariCUDA Paste Extruder.
//#define BARICUDA
```

```
/**********************************************************************\
*
* R/C SERVO support
*
* Sponsored by TrinityLabs, Reworked by codexmas
*
**********************************************************************/

// Number of servos
//
// If you select a configuration below, this will receive a default value and does not need to be set manually
// set it manually if you have more servos than extruders and wish to manually control some
// leaving it undefined or defining as 0 will disable the servo subsystem
// If unsure, leave commented / disabled
//
// #define NUM_SERVOS 3

#include "Configuration_adv.h"
#include "thermistortables.h"

#endif //__CONFIGURATION_H
```

```
#include "Marlin.h"
#include "planner.h"
#include "temperature.h"
#include "ultralcd.h"
#include "ConfigurationStore.h"


void _EEPROM_writeData(int &pos, uint8_t* value, uint8_t size)
{
  do
  {
    eeprom_write_byte((unsigned char*)pos, *value);
    pos++;
    value++;
  }while(--size);
}
#define EEPROM_WRITE_VAR(pos, value) _EEPROM_writeData(pos,
(uint8_t*)&value, sizeof(value))
void _EEPROM_readData(int &pos, uint8_t* value, uint8_t size)
{
  do
  {
    *value = eeprom_read_byte((unsigned char*)pos);
    pos++;
    value++;
  }while(--size);
}
#define EEPROM_READ_VAR(pos, value) _EEPROM_readData(pos, (uint8_t*)&value,
sizeof(value))
```

```
//============================================================================
======================
```

```
#define EEPROM_OFFSET 100
```

```
// IMPORTANT:  Whenever there are changes made to the variables  stored in EEPROM
// in the functions  below, also increment  the version  number.  This makes sure that
// the default  values  are used  whenever  there is a change  to the data, to prevent
// wrong data being  written  to the variables.
// ALSO:  always  make sure the variables  in the Store and retrieve  sections  are in the same order.
#define EEPROM_VERSION  "V07"

#ifdef EEPROM_SETTINGS
void Config_StoreSettings()
{
  char ver[4]= "000";
  int i=EEPROM_OFFSET;
  EEPROM_WRITE_VAR(i,ver);  // invalidate  data first
  EEPROM_WRITE_VAR(i,axis_steps_per_unit);
  EEPROM_WRITE_VAR(i,max_feedrate);
  EEPROM_WRITE_VAR(i,max_acceleration_units_per_sq_second);
  EEPROM_WRITE_VAR(i,acceleration);
  EEPROM_WRITE_VAR(i,retract_acceleration);
  EEPROM_WRITE_VAR(i,minimumfeedrate);
```

```
  EEPROM_WRITE_VAR(i,mintravelfeedrate);
  EEPROM_WRITE_VAR(i,minsegmenttime);
  EEPROM_WRITE_VAR(i,max_xy_jerk);
  EEPROM_WRITE_VAR(i,max_z_jerk);
  EEPROM_WRITE_VAR(i,max_e_jerk);
  EEPROM_WRITE_VAR(i,add_homeing);
  #ifndef ULTIPANEL
  int plaPreheatHotendTemp = PLA_PREHEAT_HOTEND_TEMP, plaPreheatHPBTemp
= PLA_PREHEAT_HPB_TEMP, plaPreheatFanSpeed = PLA_PREHEAT_FAN_SPEED;
  int absPreheatHotendTemp = ABS_PREHEAT_HOTEND_TEMP, absPreheatHPBTemp
= ABS_PREHEAT_HPB_TEMP, absPreheatFanSpeed = ABS_PREHEAT_FAN_SPEED;
  #endif
  EEPROM_WRITE_VAR(i,plaPreheatHotendTemp);
  EEPROM_WRITE_VAR(i,plaPreheatHPBTemp);
  EEPROM_WRITE_VAR(i,plaPreheatFanSpeed);
  EEPROM_WRITE_VAR(i,absPreheatHotendTemp);
  EEPROM_WRITE_VAR(i,absPreheatHPBTemp);
  EEPROM_WRITE_VAR(i,absPreheatFanSpeed);
  #ifdef PIDTEMP
    EEPROM_WRITE_VAR(i,Kp);
    EEPROM_WRITE_VAR(i,Ki);
    EEPROM_WRITE_VAR(i,Kd);
  #else
                float dummy = 3000.0f;
    EEPROM_WRITE_VAR(i,dummy);
                dummy = 0.0f;
    EEPROM_WRITE_VAR(i,dummy);
    EEPROM_WRITE_VAR(i,dummy);
  #endif
```

```
  char ver2[4]=EEPROM_VERSION;
  i=EEPROM_OFFSET;
  EEPROM_WRITE_VAR(i,ver2);  // validate  data
  SERIAL_ECHO_START;
  SERIAL_ECHOLNPGM("Settings  Stored");
}
#endif  //EEPROM_SETTINGS


#ifdef  EEPROM_CHITCHAT
void Config_PrintSettings()
{  // Always have this function,  even with EEPROM_SETTINGS  disabled,  the current values  will  be shown
    SERIAL_ECHO_START;
    SERIAL_ECHOLNPGM("Steps  per unit:");
    SERIAL_ECHO_START;
    SERIAL_ECHOPAIR("   M92 X",axis_steps_per_unit[0]);
    SERIAL_ECHOPAIR("  Y",axis_steps_per_unit[1]);
    SERIAL_ECHOPAIR("  Z",axis_steps_per_unit[2]);
    SERIAL_ECHOPAIR("  E",axis_steps_per_unit[3]);
    SERIAL_ECHOLN("");

    SERIAL_ECHO_START;
    SERIAL_ECHOLNPGM("Maximum  feedrates (mm/s):");
    SERIAL_ECHO_START;
    SERIAL_ECHOPAIR("   M203 X",max_feedrate[0]);
    SERIAL_ECHOPAIR("  Y",max_feedrate[1]  );
    SERIAL_ECHOPAIR("  Z", max_feedrate[2]  );
    SERIAL_ECHOPAIR("  E", max_feedrate[3]);
```

```
SERIAL_ECHOLN("");


SERIAL_ECHO_START;
SERIAL_ECHOLNPGM("Maximum Acceleration (mm/s2):");
SERIAL_ECHO_START;
SERIAL_ECHOPAIR("  M201 X" ,max_acceleration_units_per_sq_second[0] );
SERIAL_ECHOPAIR("  Y" , max_acceleration_units_per_sq_second[1] );
SERIAL_ECHOPAIR("  Z" ,max_acceleration_units_per_sq_second[2] );
SERIAL_ECHOPAIR("  E" ,max_acceleration_units_per_sq_second[3]);
SERIAL_ECHOLN("");
SERIAL_ECHO_START;
SERIAL_ECHOLNPGM("Acceleration: S=acceleration, T=retract acceleration");
SERIAL_ECHO_START;
SERIAL_ECHOPAIR("  M204 S",acceleration );
SERIAL_ECHOPAIR("  T" ,retract_acceleration);
SERIAL_ECHOLN("");


SERIAL_ECHO_START;
SERIAL_ECHOLNPGM("Advanced variables: S=Min feedrate (mm/s), T=Min travel
feedrate (mm/s), B=minimum segment time (ms), X=maximum XY jerk (mm/s),
Z=maximum Z jerk (mm/s),  E=maximum E jerk (mm/s)");
SERIAL_ECHO_START;
SERIAL_ECHOPAIR("  M205 S",minimumfeedrate );
SERIAL_ECHOPAIR("  T" ,mintravelfeedrate );
SERIAL_ECHOPAIR("  B" ,minsegmenttime );
SERIAL_ECHOPAIR("  X" ,max_xy_jerk );
SERIAL_ECHOPAIR("  Z" ,max_z_jerk);
SERIAL_ECHOPAIR("  E" ,max_e_jerk);
SERIAL_ECHOLN("");
```

```
    SERIAL_ECHO_START;
    SERIAL_ECHOLNPGM("Home offset (mm):");
    SERIAL_ECHO_START;
    SERIAL_ECHOPAIR("  M206 X",add_homeing[0] );
    SERIAL_ECHOPAIR(" Y" ,add_homeing[1] );
    SERIAL_ECHOPAIR(" Z" ,add_homeing[2] );
    SERIAL_ECHOLN("");
#ifdef PIDTEMP
    SERIAL_ECHO_START;
    SERIAL_ECHOLNPGM("PID settings:");
    SERIAL_ECHO_START;
    SERIAL_ECHOPAIR("  M301 P",Kp);
    SERIAL_ECHOPAIR(" I" ,unscalePID_i(Ki));
    SERIAL_ECHOPAIR(" D" ,unscalePID_d(Kd));
    SERIAL_ECHOLN("");
#endif
}
#endif


#ifdef EEPROM_SETTINGS
void Config_RetrieveSettings()
{
    int i=EEPROM_OFFSET;
    char stored_ver[4];
    char ver[4]=EEPROM_VERSION;
    EEPROM_READ_VAR(i,stored_ver);  //read stored version
```

```
  // SERIAL_ECHOLN("Version: [" << ver << "] Stored version: [" << stored_ver << "]");
  if (strncmp(ver,stored_ver,3) == 0)
  {
    // version number match
    EEPROM_READ_VAR(i,axis_steps_per_unit);
    EEPROM_READ_VAR(i,max_feedrate);
    EEPROM_READ_VAR(i,max_acceleration_units_per_sq_second);

    // steps per sq second need to be updated to agree with the units per sq second (as they are what is used in the planner)
        reset_acceleration_rates();

    EEPROM_READ_VAR(i,acceleration);
    EEPROM_READ_VAR(i,retract_acceleration);
    EEPROM_READ_VAR(i,minimumfeedrate);
    EEPROM_READ_VAR(i,mintravelfeedrate);
    EEPROM_READ_VAR(i,minsegmenttime);
    EEPROM_READ_VAR(i,max_xy_jerk);
    EEPROM_READ_VAR(i,max_z_jerk);
    EEPROM_READ_VAR(i,max_e_jerk);
    EEPROM_READ_VAR(i,add_homeing);
    #ifndef ULTIPANEL
    int plaPreheatHotendTemp, plaPreheatHPBTemp, plaPreheatFanSpeed;
    int absPreheatHotendTemp, absPreheatHPBTemp, absPreheatFanSpeed;
    #endif
    EEPROM_READ_VAR(i,plaPreheatHotendTemp);
    EEPROM_READ_VAR(i,plaPreheatHPBTemp);
    EEPROM_READ_VAR(i,plaPreheatFanSpeed);
```

```
      EEPROM_READ_VAR(i,absPreheatHotendTemp);
      EEPROM_READ_VAR(i,absPreheatHPBTemp);
      EEPROM_READ_VAR(i,absPreheatFanSpeed);
      #ifndef PIDTEMP
      float Kp,Ki,Kd;
      #endif
      // do not need to scale PID values as the values in EEPROM are already scaled

      EEPROM_READ_VAR(i,Kp);
      EEPROM_READ_VAR(i,Ki);
      EEPROM_READ_VAR(i,Kd);

            // Call updatePID (similar to when we have processed M301)
            updatePID();
      SERIAL_ECHO_START;
      SERIAL_ECHOLNPGM("Stored settings retrieved");
   }
   else
   {
      Config_ResetDefault();
   }
   Config_PrintSettings();
}
#endif

void Config_ResetDefault()
{
   float tmp1[]=DEFAULT_AXIS_STEPS_PER_UNIT;
   float tmp2[]=DEFAULT_MAX_FEEDRATE;
```

```
    long tmp3[]=DEFAULT_MAX_ACCELERATION;
    for (short i=0;i<4;i++)
    {
        axis_steps_per_unit[i]=tmp1[i];
        max_feedrate[i]=tmp2[i];
        max_acceleration_units_per_sq_second[i]=tmp3[i];
    }

    // steps per sq second need to be updated to agree with the units per sq second
    reset_acceleration_rates();

    acceleration=DEFAULT_ACCELERATION;
    retract_acceleration=DEFAULT_RETRACT_ACCELERATION;
    minimumfeedrate=DEFAULT_MINIMUMFEEDRATE;
    minsegmenttime=DEFAULT_MINSEGMENTTIME;
    mintravelfeedrate=DEFAULT_MINTRAVELFEEDRATE;
    max_xy_jerk=DEFAULT_XYJERK;
    max_z_jerk=DEFAULT_ZJERK;
    max_e_jerk=DEFAULT_EJERK;
    add_homeing[0] = add_homeing[1] = add_homeing[2] = 0;
#ifdef ULTIPANEL
    plaPreheatHotendTemp = PLA_PREHEAT_HOTEND_TEMP;
    plaPreheatHPBTemp = PLA_PREHEAT_HPB_TEMP;
    plaPreheatFanSpeed = PLA_PREHEAT_FAN_SPEED;
    absPreheatHotendTemp = ABS_PREHEAT_HOTEND_TEMP;
    absPreheatHPBTemp = ABS_PREHEAT_HPB_TEMP;
    absPreheatFanSpeed = ABS_PREHEAT_FAN_SPEED;
#endif
```

```
#ifdef PIDTEMP
    Kp = DEFAULT_Kp;
    Ki = scalePID_i(DEFAULT_Ki);
    Kd = scalePID_d(DEFAULT_Kd);

    // call updatePID (similar to when we have processed M301)
    updatePID();

#ifdef PID_ADD_EXTRUSION_RATE
    Kc = DEFAULT_Kc;
#endif//PID_ADD_EXTRUSION_RATE
#endif//PIDTEMP

SERIAL_ECHO_START;
SERIAL_ECHOLNPGM("Hardcoded Default Settings Loaded");

}

#ifndef CONFIG_STORE_H
#define CONFIG_STORE_H

#include "Configuration.h"

void Config_ResetDefault();

#ifdef EEPROM_CHITCHAT
void Config_PrintSettings();
#else
```

```
FORCE_INLINE void Config_PrintSettings()  {}
#endif

#ifdef EEPROM_SETTINGS
void Config_StoreSettings();
void Config_RetrieveSettings();
#else
FORCE_INLINE void Config_StoreSettings()  {}
FORCE_INLINE void Config_RetrieveSettings()  { Config_ResetDefault();
Config_PrintSettings();  }
#endif

#endif//CONFIG_STORE_H

#ifndef CONFIGURATION_ADV_H
#define CONFIGURATION_ADV_H

//===========================================================================
===========

//=============================Thermal Settings
============================

//===========================================================================
===========

#ifdef BED_LIMIT_SWITCHING
  #define BED_HYSTERESIS 2 //only disable heating if T>target+BED_HYSTERESIS
and enable heating if T>target-BED_HYSTERESIS
#endif
#define BED_CHECK_INTERVAL 5000 //ms between checks in bang-bang control
```

```
//// Heating sanity check:
// This waits for the watchperiod in milliseconds whenever an M104 or M109 increases the target temperature
// If the temperature has not increased at the end of that period, the target temperature is set to zero.
// It can be reset with another M104/M109. This check is also only triggered if the target temperature and the current temperature
// differ by at least 2x WATCH_TEMP_INCREASE
//#define WATCH_TEMP_PERIOD 40000 //40 seconds
//#define WATCH_TEMP_INCREASE 10 //Heat up at least 10 degree in 20 seconds

// Wait for Cooldown
// This defines if the M109 call should not block if it is cooling down.
// example: From a current temp of 220, you set M109 S200.
// if CooldownNoWait is defined M109 will not wait for the cooldown to finish
#define CooldownNoWait true

#ifdef PIDTEMP
 // this adds an experimental additional term to the heatingpower, proportional to the extrusion speed.
 // if Kc is choosen well, the additional required power due to increased melting should be compensated.
 #define PID_ADD_EXTRUSION_RATE
 #ifdef PID_ADD_EXTRUSION_RATE
  #define DEFAULT_Kc (1) //heatingpower=Kc*(e_speed)
 #endif
#endif
```

```
//automatic temperature: The hot end target temperature is calculated by all the buffered lines of gcode.

//The maximum buffered steps/sec of the extruder motor are called "se".

//You enter the autotemp mode by a M109 S<mintemp> T<maxtemp> F<factor>

// the target temperature is set to mintemp+factor*se[steps/sec] and limited by mintemp and maxtemp

// you exit the value by any M109 without F*

// Also, if the temperature is set to a value <mintemp, it is not changed by autotemp.

// on an ultimaker, some initial testing worked with M109 S215 B260 F1 in the start.gcode

#define AUTOTEMP

#ifdef AUTOTEMP

 #define AUTOTEMP_OLDWEIGHT 0.98

#endif


// extruder run-out prevention.

//if the machine is idle, and the temperature over MINTEMP, every couple of SECONDS some filament is extruded

//#define EXTRUDER_RUNOUT_PREVENT

#define EXTRUDER_RUNOUT_MINTEMP 190

#define EXTRUDER_RUNOUT_SECONDS 30.

#define EXTRUDER_RUNOUT_ESTEPS 14. //mm filament

#define EXTRUDER_RUNOUT_SPEED 1500. //extrusion speed

#define EXTRUDER_RUNOUT_EXTRUDE 100


//These defines help to calibrate the AD595 sensor in case you get wrong temperature measurements.

//The measured temperature is defined as "actualTemp = (measuredTemp * TEMP_SENSOR_AD595_GAIN) + TEMP_SENSOR_AD595_OFFSET"

#define TEMP_SENSOR_AD595_OFFSET 0.0

#define TEMP_SENSOR_AD595_GAIN 1.0
```

```
//This is for controlling a fan to cool down the stepper drivers
//it will turn on when any driver is enabled
//and turn off after the set amount of seconds from last driver being disabled again
#define CONTROLLERFAN_PIN -1 //Pin used for the fan to cool controller (-1 to disable)
#define CONTROLLERFAN_SECS 60 //How many seconds, after all motors were disabled, the fan should run
#define CONTROLLERFAN_SPEED 255 // == full speed

// When first starting the main fan, run it at full speed for the
// given number of milliseconds. This gets the fan spinning reliably
// before setting a PWM value. (Does not work with software PWM for fan on Sanguinololu)
//#define FAN_KICKSTART_TIME 100

// Extruder cooling fans
// Configure fan pin outputs to automatically turn on/off when the associated
// extruder temperature is above/below EXTRUDER_AUTO_FAN_TEMPERATURE.
// Multiple extruders can be assigned to the same pin in which case
// the fan will turn on when any selected extruder is above the threshold.
#define EXTRUDER_0_AUTO_FAN_PIN -1
#define EXTRUDER_1_AUTO_FAN_PIN -1
#define EXTRUDER_2_AUTO_FAN_PIN -1
#define EXTRUDER_AUTO_FAN_TEMPERATURE 50
#define EXTRUDER_AUTO_FAN_SPEED 255 // == full speed

//============================================================================
```

```
//============================Mechanical
Settings============================

//============================================================
==========


#define ENDSTOPS_ONLY_FOR_HOMING // If defined the endstops will only be used
for homing



//// AUTOSET LOCATIONS OF LIMIT SWITCHES

//// Added by ZetaPhoenix 09-15-2012

#ifdef MANUAL_HOME_POSITIONS  // Use manual limit switch locations

 #define X_HOME_POS MANUAL_X_HOME_POS

 #define Y_HOME_POS MANUAL_Y_HOME_POS

 #define Z_HOME_POS MANUAL_Z_HOME_POS

#else //Set min/max homing switch positions based upon homing direction and min/max
travel limits

 //X axis

 #if X_HOME_DIR == -1

  #ifdef BED_CENTER_AT_0_0

   #define X_HOME_POS X_MAX_LENGTH * -0.5

  #else

   #define X_HOME_POS X_MIN_POS

  #endif //BED_CENTER_AT_0_0

 #else

  #ifdef BED_CENTER_AT_0_0

   #define X_HOME_POS X_MAX_LENGTH * 0.5

  #else

   #define X_HOME_POS X_MAX_POS

  #endif //BED_CENTER_AT_0_0
```

```
  #endif //X_HOME_DIR == -1


  //Y axis
  #if Y_HOME_DIR == -1
    #ifdef BED_CENTER_AT_0_0
      #define Y_HOME_POS Y_MAX_LENGTH * -0.5
    #else
      #define Y_HOME_POS Y_MIN_POS
    #endif //BED_CENTER_AT_0_0
  #else
    #ifdef BED_CENTER_AT_0_0
      #define Y_HOME_POS Y_MAX_LENGTH * 0.5
    #else
      #define Y_HOME_POS Y_MAX_POS
    #endif //BED_CENTER_AT_0_0
  #endif //Y_HOME_DIR == -1


  // Z axis
  #if Z_HOME_DIR == -1 //BED_CENTER_AT_0_0 not used
    #define Z_HOME_POS Z_MIN_POS
  #else
    #define Z_HOME_POS Z_MAX_POS
  #endif //Z_HOME_DIR == -1
#endif //End auto min/max positions
//END AUTOSET LOCATIONS OF LIMIT SWITCHES -ZP


//#define Z_LATE_ENABLE // Enable Z the last moment. Needed if your Z driver
overheats.
```

```
// A single Z stepper driver is usually used to drive 2 stepper motors.

// Uncomment this define to utilize a separate stepper driver for each Z axis motor.

// Only a few motherboards support this, like RAMPS, which have dual extruder support (the 2nd, often unused, extruder driver is used

// to control the 2nd Z axis stepper motor). The pins are currently only defined for a RAMPS motherboards.

// On a RAMPS (or other 5 driver) motherboard, using this feature will limit you to using 1 extruder.

//#define Z_DUAL_STEPPER_DRIVERS


#ifdef Z_DUAL_STEPPER_DRIVERS

  #undef EXTRUDERS

  #define EXTRUDERS 1

#endif


//homing hits the endstop, then retracts by this distance, before it tries to slowly bump again:

#define X_HOME_RETRACT_MM 5

#define Y_HOME_RETRACT_MM 5

#define Z_HOME_RETRACT_MM 1

//#define QUICK_HOME //if this is defined, if both x and y are to be homed, a diagonal move will be performed initially.


#define AXIS_RELATIVE_MODES {false, false, false, false}


#define MAX_STEP_FREQUENCY 40000 // Max step frequency for Ultimaker (5000 pps / half step)


//By default pololu step drivers require an active high signal. However, some high power drivers require an active low signal as step.
```

```
#define INVERT_X_STEP_PIN false
#define INVERT_Y_STEP_PIN false
#define INVERT_Z_STEP_PIN false
#define INVERT_E_STEP_PIN false

//default stepper release if idle
#define DEFAULT_STEPPER_DEACTIVE_TIME 60

#define DEFAULT_MINIMUMFEEDRATE 0.0 // minimum feedrate
#define DEFAULT_MINTRAVELFEEDRATE 0.0

// minimum time in microseconds that a movement needs to take if the buffer is emptied.
#define DEFAULT_MINSEGMENTTIME 20000

// If defined the movements slow down when the look ahead buffer is only half full
#define SLOWDOWN

// Frequency limit
// See nophead's blog for more info
// Not working O
//#define XY_FREQUENCY_LIMIT 15

// Minimum planner junction speed. Sets the default minimum speed the planner plans for at the end
// of the buffer and all stops. This should not be much greater than zero and should only be changed
// if unwanted behavior is observed on a user's machine when running at very slow speeds.
#define MINIMUM_PLANNER_SPEED 0.05// (mm/sec)
```

```
// MS1 MS2 Stepper Driver Microstepping mode table

#define MICROSTEP1 LOW,LOW

#define MICROSTEP2 HIGH,LOW

#define MICROSTEP4 LOW,HIGH

#define MICROSTEP8 HIGH,HIGH

#define MICROSTEP16 HIGH,HIGH


// Microstep setting (Only functional when stepper driver microstep pins are connected to MCU.

#define MICROSTEP_MODES {16,16,16,16,16} // [1,2,4,8,16]


// Motor Current setting (Only functional when motor driver current ref pins are connected to a digital trimpot on supported boards)

#define DIGIPOT_MOTOR_CURRENT {135,135,135,135,135} // Values 0-255 (RAMBO 135 = ~0.75A, 185 = ~1A)



//===========================================================================

//=============================Additional Features=============================

//===========================================================================


#define SD_FINISHED_STEPPERRELEASE true //if sd support and the file is finished: disable steppers?

#define SD_FINISHED_RELEASECOMMAND "M84 X Y Z E" // You might want to keep the z enabled so your bed stays in place.


// The hardware watchdog should reset the Microcontroller disabling all outputs, in case the firmware gets stuck and doesn't do temperature regulation.
```

```
//#define  USE_WATCHDOG


#ifdef  USE_WATCHDOG

// If you have a watchdog reboot in an ArduinoMega2560  then the device will hang forever, as a watchdog  reset will  leave  the watchdog  on.

// The "WATCHDOG_RESET_MANUAL"  goes around this by not using  the hardware  reset.

//  However,  THIS FEATURE  IS UNSAFE!, as it will  only  work if interrupts  are disabled. And the code could hang in an interrupt  routine  with interrupts  disabled.

//#define  WATCHDOG_RESET_MANUAL

#endif


// Enable  the option  to stop SD printing  when hitting  and endstops,  needs to be enabled  from the LCD menu when this option  is enabled.

//#define  ABORT_ON_ENDSTOP_HIT_FEATURE_ENABLED


// extruder  advance  constant  (s2/mm3)

//

// advance  (steps) = STEPS_PER_CUBIC_MM_E  * EXTUDER_ADVANCE_K  * cubic  mm  per second  ^ 2

//

// hooke's law says:                force = k * distance

// bernoulli's  priniciple  says:  v ^ 2 / 2 + g . h + pressure  / density  = constant

// so: v ^ 2 is proportional  to number  of steps we advance  the extruder

//#define  ADVANCE


#ifdef  ADVANCE
  #define  EXTRUDER_ADVANCE_K  .0


  #define  D_FILAMENT  2.85
```

```
 #define STEPS_MM_E 836

 #define EXTRUTION_AREA (0.25 * D_FILAMENT * D_FILAMENT * 3.14159)

 #define STEPS_PER_CUBIC_MM_E (axis_steps_per_unit[E_AXIS]/
EXTRUTION_AREA)


#endif // ADVANCE


// Arc interpretation settings:

#define MM_PER_ARC_SEGMENT 1

#define N_ARC_CORRECTION 25


const unsigned int dropsegments=5; //everything with less than this number of steps will be ignored as move and joined with the next movement


// If you are using a RAMPS board or cheap E-bay purchased boards that do not detect when an SD card is inserted

// You can get round this by connecting a push button or single throw switch to the pin defined as SDCARDCARDDETECT

// in the pins.h file. When using a push button pulling the pin to ground this will need inverted. This setting should

// be commented out otherwise

#define SDCARDDETECTINVERTED


#ifdef ULTIPANEL

 #undef SDCARDDETECTINVERTED

#endif


// Power Signal Control Definitions

// By default use ATX definition

#ifndef POWER_SUPPLY
```

```
 #define POWER_SUPPLY 1
#endif
// 1 = ATX
#if (POWER_SUPPLY == 1)
 #define PS_ON_AWAKE LOW
 #define PS_ON_ASLEEP HIGH
#endif
// 2 = X-Box 360 203W
#if (POWER_SUPPLY == 2)
 #define PS_ON_AWAKE HIGH
 #define PS_ON_ASLEEP LOW
#endif

//===========================================================================
===========

//=============================Buffers
============================

//===========================================================================
===========

// The number of linear motions that can be in the plan at any give time.
// THE BLOCK_BUFFER_SIZE NEEDS TO BE A POWER OF 2, i.g. 8,16,32 because
shifts and ors are used to do the ringbuffering.
#if defined SDSUPPORT
 #define BLOCK_BUFFER_SIZE 16 // SD,LCD,Buttons take more memory, block
buffer needs to be smaller
#else
 #define BLOCK_BUFFER_SIZE 16 // maximize block buffer
#endif
```

```
//The ASCII buffer for recieving from the serial:
#define MAX_CMD_SIZE 96
#define BUFSIZE 4


// Firmware based and LCD controled retract
// M207 and M208 can be used to define parameters for the retraction.
// The retraction can be called by the slicer using G10 and G11
// until then, intended retractions can be detected by moves that only extrude and the direction.
// the moves are than replaced by the firmware controlled ones.

// #define FWRETRACT //ONLY PARTIALLY TESTED
#define MIN_RETRACT 0.1 //minimum extruded mm to accept a automatic gcode retraction attempt


//adds support for experimental filament exchange support M600; requires display
#ifdef ULTIPANEL
 //#define FILAMENTCHANGEENABLE
 #ifdef FILAMENTCHANGEENABLE
  #define FILAMENTCHANGE_XPOS 3
  #define FILAMENTCHANGE_YPOS 3
  #define FILAMENTCHANGE_ZADD 10
  #define FILAMENTCHANGE_FIRSTRETRACT -2
  #define FILAMENTCHANGE_FINALRETRACT -100
 #endif
#endif
```

```
//===========================================================================
==========

//============================  Define  Defines
============================

//===========================================================================
==========


#if TEMP_SENSOR_0 > 0
 #define  THERMISTORHEATER_0  TEMP_SENSOR_0
 #define  HEATER_0_USES_THERMISTOR
#endif
#if TEMP_SENSOR_1 > 0
 #define  THERMISTORHEATER_1  TEMP_SENSOR_1
 #define  HEATER_1_USES_THERMISTOR
#endif
#if TEMP_SENSOR_2 > 0
 #define  THERMISTORHEATER_2  TEMP_SENSOR_2
 #define  HEATER_2_USES_THERMISTOR
#endif
#if TEMP_SENSOR_BED  > 0
 #define  THERMISTORBED  TEMP_SENSOR_BED
 #define  BED_USES_THERMISTOR
#endif
#if TEMP_SENSOR_0 == -1
 #define  HEATER_0_USES_AD595
#endif
#if TEMP_SENSOR_1 == -1
 #define  HEATER_1_USES_AD595
```

```
#endif
#if TEMP_SENSOR_2 == -1
 #define HEATER_2_USES_AD595
#endif
#if TEMP_SENSOR_BED == -1
 #define BED_USES_AD595
#endif
#if TEMP_SENSOR_0 == -2
 #define HEATER_0_USES_MAX6675
#endif
#if TEMP_SENSOR_0 == 0
 #undef HEATER_0_MINTEMP
 #undef HEATER_0_MAXTEMP
#endif
#if TEMP_SENSOR_1 == 0
 #undef HEATER_1_MINTEMP
 #undef HEATER_1_MAXTEMP
#endif
#if TEMP_SENSOR_2 == 0
 #undef HEATER_2_MINTEMP
 #undef HEATER_2_MAXTEMP
#endif
#if TEMP_SENSOR_BED == 0
 #undef BED_MINTEMP
 #undef BED_MAXTEMP
#endif
```

```
#endif //__CONFIGURATION_ADV_H

#define START_BMPWIDTH 60 //Width in pixels
#define START_BMPHEIGHT 64 //Height in pixels
#define START_BMPBYTEWIDTH 8 //Width in bytes
const unsigned char start_bmp[574] PROGMEM = { //AVR-GCC, WinAVR
0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xF0,
0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xF0,
0xFF,0xFF,0xFF,0xF9,0xFF,0xFF,0xFF,0xF0,
0xFF,0xFF,0xFF,0xF0,0xFF,0xFF,0xFF,0xF0,
0xFF,0xFF,0xFF,0xE0,0x7F,0xFF,0xFF,0xF0,
0xFF,0xFF,0xFF,0xC0,0x3F,0xFF,0xFF,0xF0,
0xFF,0xFF,0xFF,0x80,0x1F,0xFF,0xFF,0xF0,
0xFF,0xFF,0xFF,0x00,0x0F,0xFF,0xFF,0xF0,
0xFF,0xFF,0xFE,0x00,0x07,0xFF,0xFF,0xF0,
0xFF,0xFF,0xFC,0x00,0x03,0xFF,0xFF,0xF0,
0xFF,0xFF,0xF8,0x00,0x01,0xFF,0xFF,0xF0,
0xFF,0xFF,0xF0,0x00,0x00,0xFF,0xFF,0xF0,
0xFF,0xFF,0xE0,0x00,0x00,0x7F,0xFF,0xF0,
0xFF,0xFF,0xC0,0x00,0x00,0x3F,0xFF,0xF0,
0xFF,0xFF,0x80,0x00,0x00,0x3F,0xFF,0xF0,
0xFF,0xFF,0x00,0x00,0x00,0x1F,0xFF,0xF0,
0xFF,0xFE,0x00,0x00,0x00,0x0F,0xFF,0xF0,
0xFF,0xFE,0x00,0x00,0x00,0x07,0xFF,0xF0,
0xFF,0xFC,0x00,0x00,0x00,0x07,0xFF,0xF0,
0xFF,0xFC,0x00,0x00,0x00,0x03,0xFF,0xF0,
0xFF,0xF8,0x00,0x00,0x00,0x03,0xFF,0xF0,
0xFF,0xF8,0x00,0x00,0x00,0x03,0xFF,0xF0,
```

0xFF,0xF8,0x00,0x00,0x00,0x01,0xFF,0xF0,

0xFF,0xF8,0x00,0x00,0x00,0x01,0xFF,0xF0,

0xFF,0xF8,0x00,0x00,0x00,0x01,0xFF,0xF0,

0xFF,0xF8,0x00,0x00,0x00,0x01,0xFF,0xF0,

0xFF,0xF8,0x00,0x00,0x00,0x01,0xFF,0xF0,

0xFF,0xF8,0x00,0x00,0x00,0x01,0xFF,0xF0,

0xFF,0xF8,0x00,0x00,0x00,0x01,0xFF,0xF0,

0xFF,0xF8,0x00,0x00,0x00,0x01,0xFF,0xF0,

0xFF,0xF8,0x00,0x00,0x00,0x03,0xFF,0xF0,

0xFF,0xF8,0x00,0x00,0x00,0x03,0xFF,0xF0,

0xFF,0xFC,0x00,0x00,0x00,0x03,0xFF,0xF0,

0xFF,0xFC,0x00,0x00,0x00,0x07,0xFF,0xF0,

0xFF,0xFE,0x00,0x00,0x00,0x07,0xFF,0xF0,

0xFF,0xFE,0x00,0x00,0x00,0x0F,0xFF,0xF0,

0xFF,0xFF,0x00,0x00,0x00,0x1F,0xFF,0xF0,

0xFF,0xFF,0x80,0x00,0x00,0x1F,0xFF,0xF0,

0xFF,0xFF,0xC0,0x00,0x00,0x3F,0xFF,0xF0,

0xFF,0xFF,0xE0,0x00,0x00,0x7F,0xFF,0xF0,

0xFF,0xFF,0xF0,0x00,0x01,0xFF,0xFF,0xF0,

0xFF,0xFF,0xFC,0x00,0x03,0xFF,0xFF,0xF0,

0xFF,0xFF,0xFF,0x00,0x1F,0xFF,0xFF,0xF0,

0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xF0,

0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xF0,

0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xF0,

0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xF0,

0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xF0,

0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xF0,

0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xF0,

```
0x83,0xFF,0xFF,0xFE,0x0F,0xFF,0xFF,0xF0,
0x80,0xFF,0xFF,0xFE,0x03,0xFF,0xFF,0xF0,
0x88,0x7F,0xFF,0xFE,0x23,0xFF,0xFF,0xF0,
0x8C,0x70,0x38,0x0E,0x71,0x81,0xC0,0x70,
0x8C,0x60,0x38,0x0E,0x63,0x80,0xC0,0x30,
0x80,0xE3,0x19,0xC6,0x07,0xF8,0xC7,0x30,
0x80,0xE0,0x19,0xC6,0x03,0x80,0xC7,0x10,
0x8C,0x62,0x79,0xC6,0x63,0x9C,0xC7,0x30,
0x8C,0x63,0xF8,0xC6,0x71,0x18,0xC6,0x30,
0x8E,0x30,0x18,0x0E,0x71,0x80,0xC0,0x30,
0x9E,0x38,0x39,0x1E,0x79,0xC4,0xC4,0xF0,
0xFF,0xFF,0xF9,0xFF,0xFF,0xFF,0xC7,0xF0,
0xFF,0xFF,0xF9,0xFF,0xFF,0xFF,0xC7,0xF0,
0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xF0
};

#define STATUS_SCREENWIDTH          115   //Width in pixels
#define STATUS_SCREENHEIGHT         19    //Height in pixels
#define STATUS_SCREENBYTEWIDTH      15    //Width in bytes
const unsigned char status_screen0_bmp[] PROGMEM = { //AVR-GCC, WinAVR
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x7F,0xFF,0xE0,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x70,0x00,0xE0,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x63,0x0C,0x60,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x47,0x0E,0x20,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x4F,0x0F,0x20,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x5F,0x0F,0xA0,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x5E,0x07,0xA0,
0x7F,0x80,0x00,0x3F,0xC0,0x00,0x3F,0xC0,0x00,0x41,0x04,0x00,0x40,0x60,0x20,
```

```
0xFB,0xC0,0x00,0x79,0xE0,0x00,0x79,0xE0,0x00,0x20,0x82,0x00,0x40,0xF0,0x20,
0xF3,0xC0,0x00,0x76,0xE0,0x00,0x76,0xE0,0x00,0x20,0x82,0x00,0x40,0xF0,0x20,
0xEB,0xC0,0x00,0x7E,0xE0,0x00,0x7E,0xE0,0x00,0x41,0x04,0x00,0x40,0x60,0x20,
0x7B,0x80,0x00,0x3D,0xC0,0x00,0x39,0xC0,0x00,0x82,0x08,0x00,0x5E,0x07,0xA0,
0x7B,0x80,0x00,0x3B,0xC0,0x00,0x3E,0xC0,0x01,0x04,0x10,0x00,0x5F,0x0F,0xA0,
0xFB,0xC0,0x00,0x77,0xE0,0x00,0x76,0xE0,0x01,0x04,0x10,0x00,0x4F,0x0F,0x20,
0xFB,0xC0,0x00,0x70,0xE0,0x00,0x79,0xE0,0x00,0x82,0x08,0x00,0x47,0x0E,0x20,
0xFF,0xC0,0x00,0x7F,0xE0,0x00,0x7F,0xE0,0x00,0x41,0x04,0x00,0x63,0x0C,0x60,
0x3F,0x00,0x00,0x1F,0x80,0x00,0x1F,0x80,0x00,0x00,0x00,0x00,0x70,0x00,0xE0,
0x1E,0x00,0x00,0x0F,0x00,0x00,0x0F,0x00,0x01,0xFF,0xFF,0x80,0x7F,0xFF,0xE0,
0x0C,0x00,0x00,0x06,0x00,0x00,0x06,0x00,0x01,0xFF,0xFF,0x80,0x00,0x00,0x00
};

#define STATUS_SCREENWIDTH          115   //Width in pixels
#define STATUS_SCREENHEIGHT         19    //Height in pixels
#define STATUS_SCREENBYTEWIDTH      15    //Width in bytes
const unsigned char status_screen1_bmp[] PROGMEM = { //AVR-GCC, WinAVR
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x7F,0xFF,0xE0,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x70,0x00,0xE0,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x61,0xF8,0x60,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x41,0xF8,0x20,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x40,0xF0,0x20,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x40,0x60,0x20,
0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x00,0x58,0x01,0xA0,
0x7F,0x80,0x00,0x3F,0xC0,0x00,0x3F,0xC0,0x00,0x41,0x04,0x00,0x5C,0x63,0xA0,
0xFB,0xC0,0x00,0x79,0xE0,0x00,0x79,0xE0,0x00,0x20,0x82,0x00,0x5E,0xF7,0xA0,
0xF3,0xC0,0x00,0x76,0xE0,0x00,0x76,0xE0,0x00,0x20,0x82,0x00,0x5E,0xF7,0xA0,
0xEB,0xC0,0x00,0x7E,0xE0,0x00,0x7E,0xE0,0x00,0x41,0x04,0x00,0x5C,0x63,0xA0,
```

```
0x7B,0x80,0x00,0x3D,0xC0,0x00,0x39,0xC0,0x00,0x82,0x08,0x00,0x58,0x01,0xA0,
0x7B,0x80,0x00,0x3B,0xC0,0x00,0x3E,0xC0,0x01,0x04,0x10,0x00,0x40,0x60,0x20,
0xFB,0xC0,0x00,0x77,0xE0,0x00,0x76,0xE0,0x01,0x04,0x10,0x00,0x40,0xF0,0x20,
0xFB,0xC0,0x00,0x70,0xE0,0x00,0x79,0xE0,0x00,0x82,0x08,0x00,0x41,0xF8,0x20,
0xFF,0xC0,0x00,0x7F,0xE0,0x00,0x7F,0xE0,0x00,0x41,0x04,0x00,0x61,0xF8,0x60,
0x3F,0x00,0x00,0x1F,0x80,0x00,0x1F,0x80,0x00,0x00,0x00,0x00,0x70,0x00,0xE0,
0x1E,0x00,0x00,0x0F,0x00,0x00,0x0F,0x00,0x01,0xFF,0xFF,0x80,0x7F,0xFF,0xE0,
0x0C,0x00,0x00,0x06,0x00,0x00,0x06,0x00,0x01,0xFF,0xFF,0x80,0x00,0x00,0x00
};

// Tonokip RepRap firmware rewrite based off of Hydra-mmm firmware.
// Licence: GPL

#ifndef MARLIN_H
#define MARLIN_H

#define FORCE_INLINE __attribute__((always_inline)) inline

#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <inttypes.h>

#include <util/delay.h>
#include <avr/pgmspace.h>
#include <avr/eeprom.h>
#include <avr/interrupt.h>
```

```
#include "fastio.h"
#include "Configuration.h"
#include "pins.h"

#ifndef AT90USB
#define HardwareSerial_h // trick to disable the standard HWserial
#endif

#if (ARDUINO >= 100)
# include "Arduino.h"
#else
# include "WProgram.h"
 //Arduino < 1.0.0 does not define this, so we need to do it ourselfs
# define analogInputToDigitalPin(p) ((p) + A0)
#endif

#include "MarlinSerial.h"

#ifndef cbi
#define cbi(sfr, bit) (_SFR_BYTE(sfr) &= ~_BV(bit))
#endif
#ifndef sbi
#define sbi(sfr, bit) (_SFR_BYTE(sfr) |= _BV(bit))
#endif

#include "WString.h"
```

```
#ifdef AT90USB
 #define MYSERIAL Serial
#else
 #define MYSERIAL MSerial
#endif

#define SERIAL_PROTOCOL(x) MYSERIAL.print(x);
#define SERIAL_PROTOCOL_F(x,y) MYSERIAL.print(x,y);
#define SERIAL_PROTOCOLPGM(x) serialprintPGM(PSTR(x));
#define SERIAL_PROTOCOLLN(x) {MYSERIAL.print(x);MYSERIAL.write('\n');}
#define SERIAL_PROTOCOLLNPGM(x)
{serialprintPGM(PSTR(x));MYSERIAL.write('\n');}


const char errormagic[] PROGMEM ="Error:";
const char echomagic[] PROGMEM ="echo:";
#define SERIAL_ERROR_START serialprintPGM(errormagic);
#define SERIAL_ERROR(x) SERIAL_PROTOCOL(x)
#define SERIAL_ERRORPGM(x) SERIAL_PROTOCOLPGM(x)
#define SERIAL_ERRORLN(x) SERIAL_PROTOCOLLN(x)
#define SERIAL_ERRORLNPGM(x) SERIAL_PROTOCOLLNPGM(x)

#define SERIAL_ECHO_START serialprintPGM(echomagic);
#define SERIAL_ECHO(x) SERIAL_PROTOCOL(x)
#define SERIAL_ECHOPGM(x) SERIAL_PROTOCOLPGM(x)
#define SERIAL_ECHOLN(x) SERIAL_PROTOCOLLN(x)
#define SERIAL_ECHOLNPGM(x) SERIAL_PROTOCOLLNPGM(x)
```

```
#define SERIAL_ECHOPAIR(name,value) (serial_echopair_P(PSTR(name),(value)))

void serial_echopair_P(const char *s_P, float v);
void serial_echopair_P(const char *s_P, double v);
void serial_echopair_P(const char *s_P, unsigned long v);


//things to write to serial from Programmemory. saves 400 to 2k of RAM.
FORCE_INLINE void serialprintPGM(const char *str)
{
  char ch=pgm_read_byte(str);
  while(ch)
  {
    MYSERIAL.write(ch);
    ch=pgm_read_byte(++str);
  }
}


void get_command();
void process_commands();

void manage_inactivity();

#if defined(X_ENABLE_PIN) && X_ENABLE_PIN > -1
  #define enable_x() WRITE(X_ENABLE_PIN, X_ENABLE_ON)
  #define disable_x() WRITE(X_ENABLE_PIN,!X_ENABLE_ON)
#else
```

```
  #define enable_x() ;
  #define disable_x() ;
#endif

#if defined(Y_ENABLE_PIN) && Y_ENABLE_PIN > -1
  #define enable_y() WRITE(Y_ENABLE_PIN, Y_ENABLE_ON)
  #define disable_y() WRITE(Y_ENABLE_PIN,!Y_ENABLE_ON)
#else
  #define enable_y() ;
  #define disable_y() ;
#endif

#if defined(Z_ENABLE_PIN) && Z_ENABLE_PIN > -1
  #ifdef Z_DUAL_STEPPER_DRIVERS
    #define enable_z() { WRITE(Z_ENABLE_PIN, Z_ENABLE_ON);
WRITE(Z2_ENABLE_PIN, Z_ENABLE_ON); }
    #define disable_z() { WRITE(Z_ENABLE_PIN,!Z_ENABLE_ON);
WRITE(Z2_ENABLE_PIN,!Z_ENABLE_ON); }
  #else
    #define enable_z() WRITE(Z_ENABLE_PIN, Z_ENABLE_ON)
    #define disable_z() WRITE(Z_ENABLE_PIN,!Z_ENABLE_ON)
  #endif
#else
  #define enable_z() ;
  #define disable_z() ;
#endif

#if defined(E0_ENABLE_PIN) && (E0_ENABLE_PIN > -1)
  #define enable_e0() WRITE(E0_ENABLE_PIN, E_ENABLE_ON)
```

```
  #define disable_e0() WRITE(E0_ENABLE_PIN,!E_ENABLE_ON)
#else
  #define enable_e0()  /* nothing */
  #define disable_e0() /* nothing */
#endif

#if (EXTRUDERS > 1) && defined(E1_ENABLE_PIN) && (E1_ENABLE_PIN > -1)
  #define enable_e1() WRITE(E1_ENABLE_PIN, E_ENABLE_ON)
  #define disable_e1() WRITE(E1_ENABLE_PIN,!E_ENABLE_ON)
#else
  #define enable_e1()  /* nothing */
  #define disable_e1() /* nothing */
#endif

#if (EXTRUDERS > 2) && defined(E2_ENABLE_PIN) && (E2_ENABLE_PIN > -1)
  #define enable_e2() WRITE(E2_ENABLE_PIN, E_ENABLE_ON)
  #define disable_e2() WRITE(E2_ENABLE_PIN,!E_ENABLE_ON)
#else
  #define enable_e2()  /* nothing */
  #define disable_e2() /* nothing */
#endif


enum AxisEnum {X_AXIS=0, Y_AXIS=1, Z_AXIS=2, E_AXIS=3};


void FlushSerialRequestResend();
void ClearToSend();
```

```
void get_coordinates();
void prepare_move();
void kill();
void Stop();

bool IsStopped();

void enquecommand(const char *cmd); //put an ascii command at the end of the current buffer.
void enquecommand_P(const char *cmd); //put an ascii command at the end of the current buffer, read from flash
void prepare_arc_move(char isclockwise);
void clamp_to_software_endstops(float target[3]);

#ifdef FAST_PWM_FAN
void setPwmFrequency(uint8_t pin, int val);
#endif

#ifndef CRITICAL_SECTION_START
 #define CRITICAL_SECTION_START  unsigned char _sreg = SREG; cli();
 #define CRITICAL_SECTION_END    SREG = _sreg;
#endif //CRITICAL_SECTION_START

extern float homing_feedrate[];
extern bool axis_relative_modes[];
extern int feedmultiply;
extern int extrudemultiply; // Sets extrude multiply factor (in percent)
extern float current_position[NUM_AXIS] ;
```

```
extern float add_homeing[3];
extern float min_pos[3];
extern float max_pos[3];
extern int fanSpeed;
#ifdef BARICUDA
extern int ValvePressure;
extern int EtoPPressure;
#endif

#ifdef FWRETRACT
extern bool autoretract_enabled;
extern bool retracted;
extern float retract_length, retract_feedrate, retract_zlift;
extern float retract_recover_length, retract_recover_feedrate;
#endif

extern unsigned long starttime;
extern unsigned long stoptime;

// Handling multiple extruders pins
extern uint8_t active_extruder;

#endif
/*
  HardwareSerial.cpp - Hardware serial library for Wiring
  Copyright (c) 2006 Nicholas Zambetti. All right reserved.

  This library is free software; you can redistribute it and/or
```

```
  modify it under the terms of the GNU Lesser General Public
  License as published by the Free Software Foundation; either
  version 2.1 of the License, or (at your option) any later version.

  This library is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
  Lesser General Public License for more details.

  You should have received a copy of the GNU Lesser General Public
  License along with this library; if not, write to the Free Software
  Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA

  Modified 23 November 2006 by David A. Mellis
  Modified 28 September 2010 by Mark Sproul
*/

#include "Marlin.h"
#include "MarlinSerial.h"

#ifndef AT90USB
// this next line disables the entire HardwareSerial.cpp,
// this is so I can support Attiny series and any other chip without a uart
#if defined(UBRRH) || defined(UBRR0H) || defined(UBRR1H) || defined(UBRR2H) ||
defined(UBRR3H)

#if UART_PRESENT(SERIAL_PORT)
  ring_buffer rx_buffer  = { { 0 }, 0, 0 };
#endif
```

```
FORCE_INLINE void store_char(unsigned char c)
{
  int i = (unsigned int)(rx_buffer.head + 1) % RX_BUFFER_SIZE;

  // if we should be storing the received character into the location
  // just before the tail (meaning that the head would advance to the
  // current location of the tail), we're about to overflow the buffer
  // and so we don't write the character or advance the head.
  if (i != rx_buffer.tail) {
    rx_buffer.buffer[rx_buffer.head] = c;
    rx_buffer.head = i;
  }
}


//#elif defined(SIG_USART_RECV)
#if defined(M_USARTx_RX_vect)
  // fixed by Mark Sproul this is on the 644/644p
  //SIGNAL(SIG_USART_RECV)
  SIGNAL(M_USARTx_RX_vect)
  {
    unsigned char c = M_UDRx;
    store_char(c);
  }
#endif

// Constructors ////////////////////////////////////////////////////////////////
```

```
MarlinSerial::MarlinSerial()
{

}

// Public Methods //////////////////////////////////////////////////////////////

void MarlinSerial::begin(long baud)
{
  uint16_t baud_setting;
  bool useU2X = true;

#if F_CPU == 16000000UL && SERIAL_PORT == 0
  // hardcoded exception for compatibility with the bootloader shipped
  // with the Duemilanove and previous boards and the firmware on the 8U2
  // on the Uno and Mega 2560.
  if (baud == 57600) {
    useU2X = false;
  }
#endif

  if (useU2X) {
    M_UCSRxA = 1 << M_U2Xx;
    baud_setting = (F_CPU / 4 / baud - 1) / 2;
  } else {
    M_UCSRxA = 0;
    baud_setting = (F_CPU / 8 / baud - 1) / 2;
```

```
  }

  // assign the baud_setting, a.k.a. ubbr (USART Baud Rate Register)
  M_UBRRxH = baud_setting >> 8;
  M_UBRRxL = baud_setting;

  sbi(M_UCSRxB, M_RXENx);
  sbi(M_UCSRxB, M_TXENx);
  sbi(M_UCSRxB, M_RXCIEx);
}

void MarlinSerial::end()
{
  cbi(M_UCSRxB, M_RXENx);
  cbi(M_UCSRxB, M_TXENx);
  cbi(M_UCSRxB, M_RXCIEx);
}



int MarlinSerial::peek(void)
{
  if (rx_buffer.head == rx_buffer.tail) {
    return -1;
  } else {
    return rx_buffer.buffer[rx_buffer.tail];
  }
}
```

```
int MarlinSerial::read(void)
{
  // if the head isn't ahead of the tail, we don't have any characters
  if (rx_buffer.head == rx_buffer.tail) {
    return -1;
  } else {
    unsigned char c = rx_buffer.buffer[rx_buffer.tail];
    rx_buffer.tail = (unsigned int)(rx_buffer.tail + 1) % RX_BUFFER_SIZE;
    return c;
  }
}

void MarlinSerial::flush()
{
  // don't reverse this or there may be problems if the RX interrupt
  // occurs after reading the value of rx_buffer_head but before writing
  // the value to rx_buffer_tail; the previous value of rx_buffer_head
  // may be written to rx_buffer_tail, making it appear as if the buffer
  // don't reverse this or there may be problems if the RX interrupt
  // occurs after reading the value of rx_buffer_head but before writing
  // the value to rx_buffer_tail; the previous value of rx_buffer_head
  // may be written to rx_buffer_tail, making it appear as if the buffer
  // were full, not empty.
  rx_buffer.head = rx_buffer.tail;
}
```

```
/// imports from print.h



void MarlinSerial::print(char c, int base)
{
  print((long) c, base);
}

void MarlinSerial::print(unsigned char b, int base)
{
  print((unsigned long) b, base);
}

void MarlinSerial::print(int n, int base)
{
  print((long) n, base);
}

void MarlinSerial::print(unsigned int n, int base)
{
  print((unsigned long) n, base);
}

void MarlinSerial::print(long n, int base)
```

```
{
  if (base == 0) {
    write(n);
  } else if (base == 10) {
    if (n < 0) {
      print('-');
      n = -n;
    }
    printNumber(n, 10);
  } else {
    printNumber(n, base);
  }
}

void MarlinSerial::print(unsigned long n, int base)
{
  if (base == 0) write(n);
  else printNumber(n, base);
}

void MarlinSerial::print(double n, int digits)
{
  printFloat(n, digits);
}

void MarlinSerial::println(void)
{
  print('\r');
```

```
  print('\n');
}

void MarlinSerial::println(const  String  &s)
{
  print(s);
  println();
}

void MarlinSerial::println(const  char c[])
{
  print(c);
  println();
}

void MarlinSerial::println(char  c, int base)
{
  print(c,  base);
  println();
}

void MarlinSerial::println(unsigned  char b, int base)
{
  print(b,  base);
  println();
}

void MarlinSerial::println(int  n, int base)
```

```
{
  print(n,  base);
  println();
}

void MarlinSerial::println(unsigned  int n, int base)
{
  print(n,  base);
  println();
}

void MarlinSerial::println(long  n, int base)
{
  print(n,  base);
  println();
}

void MarlinSerial::println(unsigned  long  n, int base)
{
  print(n,  base);
  println();
}

void MarlinSerial::println(double  n, int digits)
{
  print(n,  digits);
  println();
}
```

```
// Private Methods //////////////////////////////////////////////////////////

void MarlinSerial::printNumber(unsigned long n, uint8_t base)
{
  unsigned char buf[8 * sizeof(long)]; // Assumes 8-bit chars.
  unsigned long i = 0;

  if (n == 0) {
    print('0');
    return;
  }

  while (n > 0) {
    buf[i++] = n % base;
    n /= base;
  }

  for (; i > 0; i--)
    print((char) (buf[i - 1] < 10 ?
      '0' + buf[i - 1] :
      'A' + buf[i - 1] - 10));
}

void MarlinSerial::printFloat(double number, uint8_t digits)
{
  // Handle negative numbers
  if (number < 0.0)
```

```
{
  print('-');
  number = -number;
}

// Round correctly so that print(1.999, 2) prints as "2.00"
double rounding = 0.5;
for (uint8_t i=0; i<digits; ++i)
  rounding /= 10.0;

number += rounding;

// Extract the integer part of the number and print it
unsigned long int_part = (unsigned long)number;
double remainder = number - (double)int_part;
print(int_part);

// Print the decimal point, but only if there are digits beyond
if (digits > 0)
  print(".");

// Extract digits from the remainder one at a time
while (digits-- > 0)
{
  remainder *= 10.0;
  int toPrint = int(remainder);
  print(toPrint);
  remainder -= toPrint;
```

```
  }
}
// Preinstantiate Objects //////////////////////////////////////////////////

MarlinSerial MSerial;

#endif // whole file
#endif // !AT90USB
/*
  HardwareSerial.h - Hardware serial library for Wiring
  Copyright (c) 2006 Nicholas Zambetti.  All right reserved.

  This library is free software; you can redistribute it and/or
  modify it under the terms of the GNU Lesser General Public
  License as published by the Free Software Foundation; either
  version 2.1 of the License, or (at your option) any later version.

  This library is distributed in the hope that it will be useful,
  but WITHOUT ANY WARRANTY; without even the implied warranty of
  MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the GNU
  Lesser General Public License for more details.

  You should have received a copy of the GNU Lesser General Public
  License along with this library; if not, write to the Free Software
  Foundation, Inc., 51 Franklin St, Fifth Floor, Boston, MA  02110-1301  USA

  Modified 28 September 2010 by Mark Sproul
```

```
*/

#ifndef MarlinSerial_h
#define MarlinSerial_h
#include "Marlin.h"

#if !defined(SERIAL_PORT)
#define SERIAL_PORT 0
#endif

// The presence of the UBRRH register is used to detect a UART.
#define UART_PRESENT(port) ((port == 0 && (defined(UBRRH) || defined(UBRR0H))) || \
                            (port == 1 && defined(UBRR1H)) || (port == 2 && defined(UBRR2H)) || \
                            (port == 3 && defined(UBRR3H)))

// These are macros to build serial port register names for the selected SERIAL_PORT (C preprocessor
// requires two levels of indirection to expand macro values properly)
#define SERIAL_REGNAME(registerbase,number,suffix) SERIAL_REGNAME_INTERNAL(registerbase,number,suffix)
#if SERIAL_PORT == 0 && (!defined(UBRR0H) || !defined(UDR0)) // use un-numbered registers if necessary
#define SERIAL_REGNAME_INTERNAL(registerbase,number,suffix) registerbase##suffix
#else
#define SERIAL_REGNAME_INTERNAL(registerbase,number,suffix) registerbase##number##suffix
#endif
```

```
// Registers used by MarlinSerial class (these are expanded
// depending on selected serial port
#define M_UCSRxA SERIAL_REGNAME(UCSR,SERIAL_PORT,A) // defines M_UCSRxA to be UCSRnA where n is the serial port number
#define M_UCSRxB SERIAL_REGNAME(UCSR,SERIAL_PORT,B)
#define M_RXENx SERIAL_REGNAME(RXEN,SERIAL_PORT,)
#define M_TXENx SERIAL_REGNAME(TXEN,SERIAL_PORT,)
#define M_RXCIEx SERIAL_REGNAME(RXCIE,SERIAL_PORT,)
#define M_UDREx SERIAL_REGNAME(UDRE,SERIAL_PORT,)
#define M_UDRx SERIAL_REGNAME(UDR,SERIAL_PORT,)
#define M_UBRRxH SERIAL_REGNAME(UBRR,SERIAL_PORT,H)
#define M_UBRRxL SERIAL_REGNAME(UBRR,SERIAL_PORT,L)
#define M_RXCx SERIAL_REGNAME(RXC,SERIAL_PORT,)
#define M_USARTx_RX_vect SERIAL_REGNAME(USART,SERIAL_PORT,_RX_vect)
#define M_U2Xx SERIAL_REGNAME(U2X,SERIAL_PORT,)

#define DEC 10
#define HEX 16
#define OCT 8
#define BIN 2
#define BYTE 0

#ifndef AT90USB
// Define constants and variables for buffering incoming serial data. We're
// using a ring buffer (I think), in which rx_buffer_head is the index of the
```

```
// location to which to write the next incoming character and rx_buffer_tail
// is the index of the location from which to read.
#define RX_BUFFER_SIZE 128


struct ring_buffer
{
  unsigned char buffer[RX_BUFFER_SIZE];
  int head;
  int tail;
};

#if UART_PRESENT(SERIAL_PORT)
  extern ring_buffer rx_buffer;
#endif

class MarlinSerial //: public Stream
{

  public:
    MarlinSerial();
    void begin(long);
    void end();
    int peek(void);
    int read(void);
    void flush(void);

    FORCE_INLINE int available(void)
```

```
  {
    return (unsigned int)(RX_BUFFER_SIZE + rx_buffer.head - rx_buffer.tail) %
RX_BUFFER_SIZE;
  }

  FORCE_INLINE void write(uint8_t c)
  {
   while (!((M_UCSRxA) & (1 << M_UDREx)))
     ;

   M_UDRx = c;
  }



  FORCE_INLINE void checkRx(void)
  {
   if((M_UCSRxA & (1<<M_RXCx)) != 0) {
     unsigned char c = M_UDRx;
     int i = (unsigned int)(rx_buffer.head + 1) % RX_BUFFER_SIZE;

     // if we should be storing the received character into the location
     // just before the tail (meaning that the head would advance to the
     // current location of the tail), we're about to overflow the buffer
     // and so we don't write the character or advance the head.
     if (i != rx_buffer.tail) {
       rx_buffer.buffer[rx_buffer.head] = c;
       rx_buffer.head = i;
     }
   }
```

```
  }


  private:
  void printNumber(unsigned long, uint8_t);
  void printFloat(double, uint8_t);


public:

  FORCE_INLINE void write(const char *str)
  {
    while (*str)
      write(*str++);
  }


  FORCE_INLINE void write(const uint8_t *buffer, size_t size)
  {
    while (size--)
      write(*buffer++);
  }

  FORCE_INLINE void print(const String &s)
  {
    for (int i = 0; i < (int)s.length(); i++) {
      write(s[i]);
    }
```

```
    }

    FORCE_INLINE void print(const  char *str)
    {
      write(str);
    }
    void print(char,  int = BYTE);
    void print(unsigned  char, int = BYTE);
    void print(int,  int = DEC);
    void print(unsigned  int, int = DEC);
    void print(long,  int = DEC);
    void print(unsigned  long, int = DEC);
    void print(double,  int = 2);

    void println(const  String &s);
    void println(const  char[]);
    void println(char,  int = BYTE);
    void println(unsigned  char, int = BYTE);
    void println(int,  int = DEC);
    void println(unsigned  int, int = DEC);
    void println(long,  int = DEC);
    void println(unsigned  long, int = DEC);
    void println(double,  int = 2);
    void println(void);
};
extern MarlinSerial  MSerial;
#endif // !AT90USB
#endif
```