 doi:10.20944/preprints202112.0460.v1

Article

MTouch: An Automatic Fault Detection System for Desktop FFF 3D Printers using a Contact Sensor

Samuel Aidala [1], Zachary Eichenberger [1], Nicholas Chan[1], Kyle Wilkinson [1], and Chinedum Okwudire [1,*]

[1] Department of Mechanical Engineering, University of Michigan, Ann Arbor, MI 48109, USA;
* Correspondence: okwudire@umich.edu

Abstract: Desktop fused filament fabrication (FFF) 3D printers have been growing in popularity among hobbyist and professional users as a prototyping and low-volume manufacturing tool. One issue these printers face is the inability to determine when a defect has occurred rendering the print unusable. Several techniques have been proposed to detect such defects but many of these approaches are tailored to one specific fault (e.g., filament runout/jam), use expensive hardware such as laser distance sensors, and/or use machine vision algorithms which are sensitive to ambient conditions, and hence can be unreliable. This paper proposes a versatile, reliable, and low-cost system, named MTouch, to detect millimeter-scale defects that tend to make prints unusable. At the core of MTouch is an actuated contact probe designed using a low-power solenoid, magnet, and hall effect sensor. This sensor is used to check for the presence, or absence, of the printed object at specific locations. The MTouch probe demonstrated 100% reliability, which was significantly higher than the 74% reliability achieved using a commercially available contact probe (the BLTouch). Additionally, an algorithm was developed to automatically detect common print failures such as layer shifting, bed separation, and filament runout using the MTouch probe. The algorithm was implemented on a Raspberry Pi mini-computer via an Octoprint plug-in. In head-to-head testing against a commercially available print defect detection system (The Spaghetti Detective), the MTouch was able to detect faults 44% faster on average while only increasing the print time by 8.49%. In addition, MTouch was able to detect faults The Spaghetti Detective was unable to identify such as layer shifting and filament runout/jam.

Keywords: Fault Detection; 3D Printer; Error Detection; FFF; Contact Sensor

1. Introduction

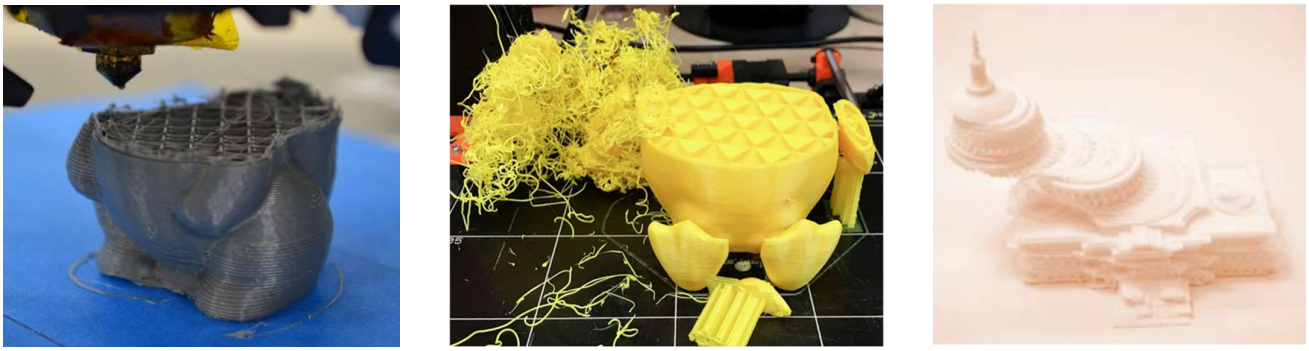
Fused filament fabrication (FFF) 3D printers, relied on by over 80% of 3D printing users [1], work by heating plastic filament and extruding it through a nozzle onto a bed. To manufacture an object, motion systems move the nozzle and bed to build up the desired part layer by layer. Desktop 3D printers, which are mostly FFF printers, are defined as printers that retail for less than $5000 [2]. Their adoption has been growing rapidly among hobbyists and professional users. In 2019, over 700,000 units of desktop 3D printers retailing at an average selling price of $1,196 were sold globally, representing a 19.4% increase over the year 2018 [2].

A hurdle in the further adoption of desktop FFF 3D printers is their failure rate of more than 20% [3]. When a defect develops, the printer will often continue manufacturing an unusable part, wasting time and material, unless manually stopped. To catch issues early, printers are often watched by a user; however, this may not always be possible. Some prints may take overnight to complete, or a user may be busy tending to another task. Similarly, for large collections of printers, often called print farms, used for small scale production, watching numerous printers can become tiring and an operator may not notice one printer with an error among many others. An automated fault detection system eliminates the need for constant supervision when printing, allowing print farm operators

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 December 2021 doi:10.20944/preprints202112.0460.v1

to attend to other tasks and personal users to leave printers operating unsupervised with greater peace of mind.

One issue in the development of automated fault detection systems is the wide variety of faults that can occur. Defects in a print can range from overly rough surface finish but an otherwise geometrically acceptable part, to prints with layers shifted above a certain height, to collapsed sections of a print resulting in an unusable part. For the purposes of this paper, major print detects are defined as faults which cause millimeter scale defects resulting in a print being unsuitable for use in its intended application. Examples of major faults which this paper will specifically focus on are bed separation, filament runout/jams, and layer shifting [4]. Some examples of these faults can be seen in Figure 1. In addition to these defects, another type of defect will be mentioned, spaghetti failure. This fault it considered a secondary fault, as it occurs as the result of another major fault (the primary fault) such as bed separation or partial collapse of the print [5].



(**a**) (**b**) (**c**)

Figure 1. Examples of major faults including: (**a**) Filament Runout/Jam, (**b**) Spaghetti Failure, (**c**) Layer Shift [4,6,7]

A number of academic papers have been published on potential in situ fault detection systems for FFF 3D using a variety of sensors [8]. Fault detection system using optical cameras has been an active area of research [9-14]. How these systems work can be divided into two main categories. The first category uses multiple cameras and feature extraction to create a point cloud representing the partially completed print. This point cloud is then either directly compared to a point cloud generated using the STL file of the desired part [9] or is input into machine learning algorithms to determine when a fault has occurred [10] (some machine learning algorithms also take in the point cloud from the STL file [11]). The second category of fault detection system inputs the image stream from the optical camera directly into a ML algorithm [12] (most commonly a neural network [13,14]).

One system commercially available for major fault detection which implements this second category is The Spaghetti Detective [15]. This system uses a Raspberry Pi and a web camera aimed at the side of the print at roughly the level of the bed. The camera streams a video feed of the print to either a separate computer set up by the user, executing the open-source project code, or to a server managed by the company overseeing the project and offering the service commercially. This separate machine uses a trained neural network to process the video feed and alerts the users if any faults are detected. The Spaghetti Detective does have some limitations. Accurate fault detection requires consistent lighting and a plain background to prevent false positives [15]. Another limitation is that, while the system can detect spaghetti failures, it is often unable to detect faults which do not result in spaghetti failure such as layer shifting and under-/over-extrusion [16]. Research groups have been able to improve this limitation by training on larger datasets to include under-extrusion and over-extrusion [17] but large geometric defects like layer shifting are still often unable to be detected. The first category of camera based systems perform better when detecting these geometric defects but still require consistent lighting

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 December 2021 doi:10.20944/preprints202112.0460.v1

and a plain background. These systems also require multiple cameras which must be calibrated and more precisely positioned [11].

Another active area of interest regarding fault detection is using laser scanners [18-20]. Systems based on these laser scanners operate in a similar manner to those in the first category of optical cameras but use laser scanners to create a point cloud of the partially completed print. Khanzadeh et al. [18] developed a laser scanning fault detection system which compares the measured point cloud to one generated from the STL model of the desired part. Tootooni et al. [19] compares point clouds with the use of a trained neural network. A benefit these systems have over optical camera-based systems is robustness against environmental factors. Ambient lighting conditions have much less of an effect on laser sensors and do not require a plain background for accurate measurements. These systems are not without their downsides. The laser scanners can be affected by laser shadowing causing some parts of the print to be unmeasurable. Additionally, the cost of the laser scanners used in many of these systems are of considerable price, often costing multiple hundreds of dollars for the scanner and more for the controlling/interfacing hardware necessary to operate them.

In addition to these optical camera and laser scan detection systems, systems have also been designed using accelerometers [20], piezoelectric vibration sensors [21], microscopes [22], borescopes [23], and acoustic emission sensor [24]. While these alternatives can offer ranges of benefits from being low cost, to robust against environmental factors, these systems only focus on detecting surface roughness, nozzle clogging, and interlayer bonding. Larger scale geometric defects are likely to be undetected by these systems as they are designed with a different goal. Some printers also come with a filament sensor to detect when the printer's filament spool has run out. While these sensors can detect when the filament runs out (as the name implies), they cannot determine when the nozzle is clogged or when geometric defects, like layer shifting, occur.

Therefore, there is a gap in the current research for a fault detection system that is low cost, robust against ambient conditions, and able to detect millimeter-scale geometric defects which often ruin prints completely. In this paper, we present such an automatic fault detection system, called MTouch. The proposed system is based on using an actuated contact sensor to detect the presence or absence of a print at predetermined locations across layers of an object being printed. The effectiveness of MTouch is shown through testing and shown to have advantages over The Spaghetti Detective such as being able to detect layer shifting and filament runout/jam faults. The outline of the rest of this article is as follows: Sections 2.1-2.3 describe the MTouch system methodology. This includes how faults are detected, sample points are generated, and the development of a sensor to detect the presence or absence of the print. Section 2.4 provides an overview of the experimental setup used in the validations tests of the sensor and system described in Section 3.1. Section 3.2 presents the results of these tests before conclusions are drawn in Section 4 showing the effectiveness of the MTouch system and future work is discussed.

2. Materials and Methods

2.1. Detecting Faults

MTouch is an automatic fault detection system which works on the principle of determining if the object being printed has millimeter-scale geometric defects by determining the present or absence of the print at specific locations and comparing to precomputed expected results. MTouch consists of two main components: the controlling software and the sensor. A high-level flowchart of the entire MTouch system can be seen in Figure 2. The system starts by generating sample points and their expected results from a Gcode file and user inputs. Next, the printing process is interrupted after a layer with generated sample points is completed. During the interruption, the sensor, is used to determine if the printed object is present or absent at each location. The presence or absence of the print is determined by actuating a contact sensor and recording if contact was made with the print. The results of these measurements are compared to the pre-

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 December 2021 doi:10.20944/preprints202112.0460.v1

computed expected result. If all results match their expected outcomes, the print is resumed. If there is a discrepancy, the print is paused, and the user is alerted to a potential fault.

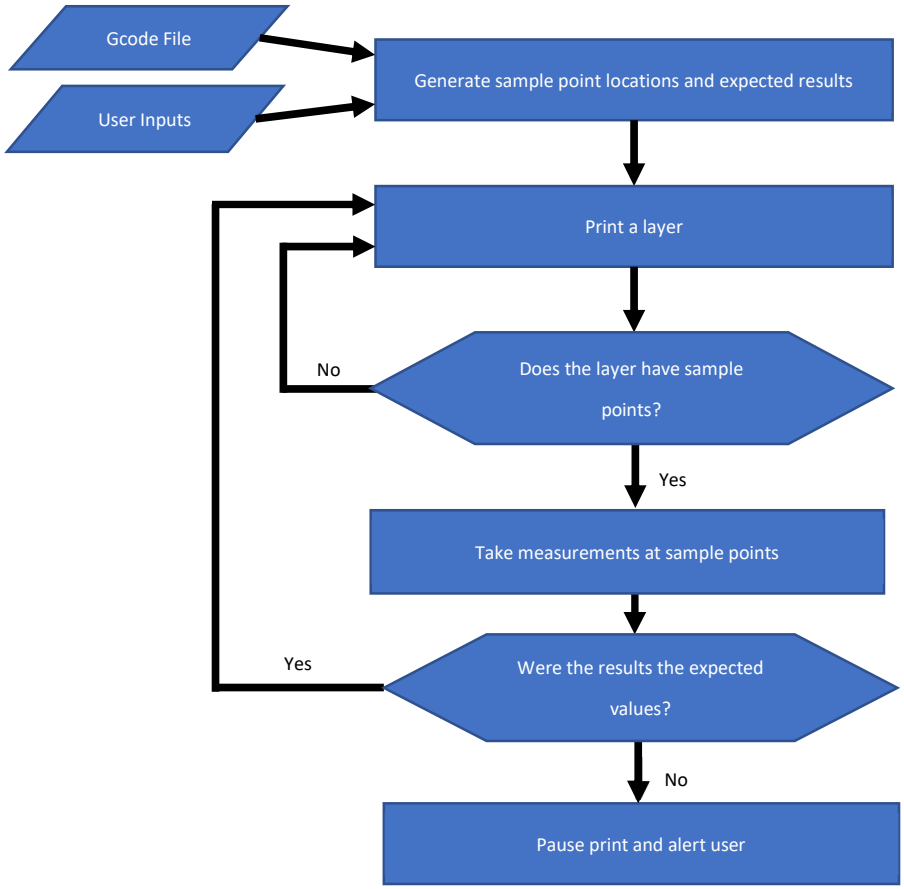


Figure 2. A flowchart illustrating the MTouch system at a high level.

2.2. Control Software

The first responsibility of the control software is all the pre-processing work. This includes creating a model of the print and using this model to generate the location of sample points as well as their expected results. The control software is also responsible for handling key events during the printing process, including interrupting the print at the layers to be sampled, moving the extruder to position the sensor over the sample point, using the sensor to get a measurement, and processing the result. Figure 3 shows a flowchart of these processes handled by the control software. Each process is covered in greater detail in the subsequent subsections below.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 December 2021 doi:10.20944/preprints202112.0460.v1

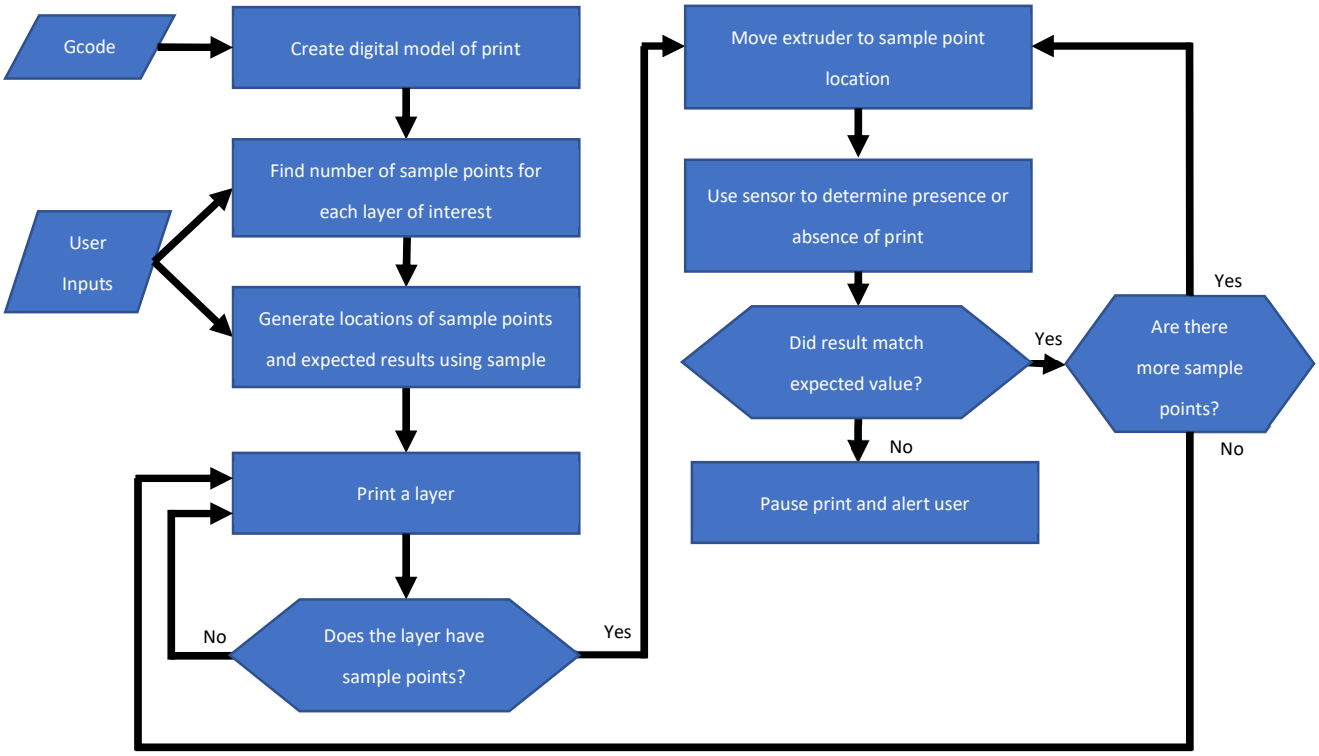


Figure 3. A flowchart illustrating the fault detection process at a high level

2.2.1. Print Model

The control software starts by creating a model of the print, used for sample point generation, from the input Gcode file. This model is created by representing a layer as a set of points whose coordinates correspond to the values of the X, Y, and Z components of G0/G1 commands (commands with no E component are ignored as nothing is being printed). To ensure a dense enough set of points which accurately represent a layer, extra points are added to commands with lengths over 0.1 mm to ensure points were no further away than 0.1 mm. The value of 0.1 mm was chosen because it is equal to the X/Y accuracy of the Ender 3 Pro printer being used for validation testing. Additionally, the smallest features the sensor can detect is limited by the size of the probe tip which is significantly larger than 0.1mm, and therefore any finer resolution could not be detected.

2.2.2. User Inputs

After generating a model of the print, the control software takes in some inputs from the user. The first input is the X, Y and Z offsets of the sensor's probe tip from the extruder nozzle (the Z distance represents the distance the probe tip extends past the extruder at full extension). Figure 4 shows an illustration of a side view of the extruder and sensor probe tip where the X/Y and Z offsets are indicated. This input is used to calculate where to position the extruder so the sensor is over the intended sample point.

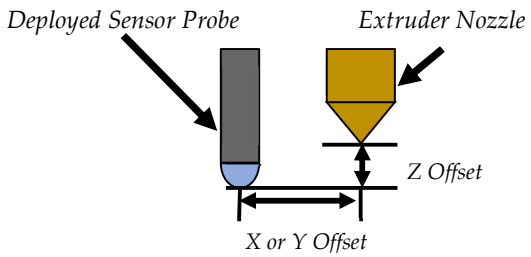


Figure 4. The X/Y and Z offsets of the sensor probe tip from the extruder nozzle

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 December 2021 doi:10.20944/preprints202112.0460.v1

In addition to these offsets, a filament retraction value is input so the software knows how far the filament should be retracted to prevent plastic from seeping out of the nozzle during the sampling process. The next input is the layer spacing. This is the interval between layers that are sampled (e.g. for a layer spacing of 5, sample points are generated for every 5th layer). The smaller the layer spacing the more quickly a fault will be detected but the more time will be taken measuring sample points. Finally, the control software takes in which algorithms to use when generating sample points and the area density of the points to generate by each algorithm. Four algorithms were developed as detailed in the next section. Multiple can be selected and the generated points combined. Point density is used instead of a desired number of points to account for changing cross sectional area of each layer. This saves time by preventing smaller layers from being oversampled with many redundant points.

2.2.3. Sample Point Generation

Sample points are created by iterating through the layers of the model and generating sample points at the interval specified by the layer spacing. Special care must be taken with sample points generated for layers with heights less than the Z offset of the probe. This is because, if the probe were to not contact the print, it will, instead, contact the bed of the printer before it reaches full extension and would register as a "contact" result causing a false reading. To resolve this issue, the extruder must be raised until the probe no longer contacts the bed at full extension.

Figure 5 shows some example sample points generated for layers of a test print. At locations like the shell of a print or an infill wall, the print should be detected. Similarly, at other locations, the print is not expected to be found such as slightly beyond the outer shell of a print or cavities within the print. If a fault were to occur during the printing process, the result of a sample point will not match its expected outcome. For example, if a filament runout fault were to occur, sample points tested after the filament has run out will return a "no print detected" result despite expecting the print to be detected. When this discrepancy is found, the print can be paused, and the user alerted to the issue.

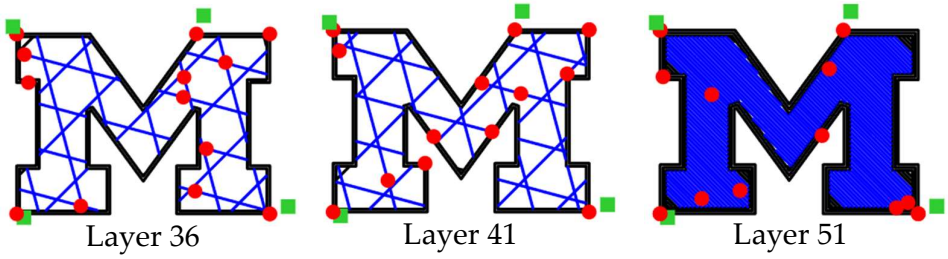


Figure 5. Sample points generated for three different layers of a test print. Points where the print is expected to be detected are shown using red circles; points where the print is not expected are shown using green squares.

Sample points for each layer are generated using strategies that are picked and combined by the user. For each layer of interest, the number of sample points generated with each strategy is first calculated by using the area of the convex hull of the layer and point density specified for each specific strategy input by the user. Each sampling strategy is then used to calculate the specified number of sample points before moving to the next layer. One possible strategy is random placement where random points of the layer's model are selected. Since these points are of the layer itself, the expected result for each one is a "contact" reading.

Beyond the random placement strategy, three advanced sampling strategies were developed to optimize the location of sample points to maximize the chances of detecting faults. The first sampling strategy developed was the Inside-Outside strategy. This strategy aims to ensure sample points are generated along the outer shell of the print by finding the convex hull of the cross section and generating points within a 0.1 mm threshold. The same number of points are then generated outside this threshold and

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 December 2021 doi:10.20944/preprints202112.0460.v1

within the convex hull to ensure the internal parts of the print are also sampled. Figure 6(a) shows sample points generated using the Inside-Outside strategy for a test print. By sampling along the convex hull, faults are more likely to be detected. This is due to any fault causing the shell to shift toward the center of the print resulting in a "No print detected" result when the print was expected to be found. Additionally, other faults such as collapsed overhangs and misshapen shells are most likely to be along the convex hull of a print, where other parts of the print cannot provide support.

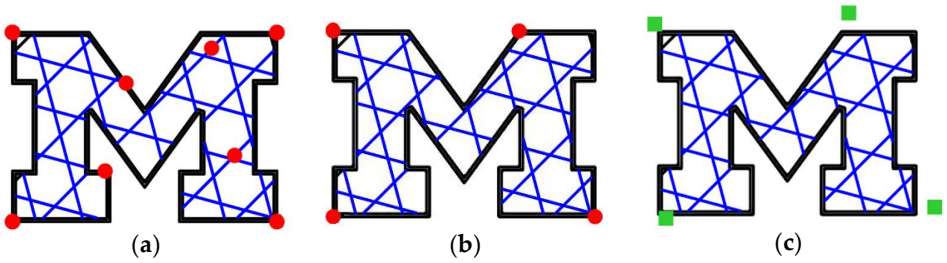


Figure 6. Sample points generated with each sampling strategy: **(a)** Inside-Outside; **(b)** Min-Max; **(c)** Layer Shifting Detection

The second sampling strategy developed was the Min-Max strategy. This process calculates points at the extremes of the X and Y axes. Figure 6(b) exhibits sample points output by Min-Max for a test print. By taking sample points at the extremes of the X and Y axes, faults that allow the print to move relative to the extruder will be reliably detected. An example of such a fault is bed separation. Since the generated sample points are at the extremes along the X and Y axes, at least one of the sample points will no longer be aligned with the print as it slides on the bed after separation. This sampling strategy also ensures that some sample points are spread across the entire layer and do not become concentrated in one area by chance as points are randomly chosen.

Note, the Min-Max strategy is not able to detect layer shift faults. This is because the layers after the shift and the extruder will both move the same amount. Since there is no net difference, the sample points will still detect the print where expected. This led to the final sampling strategy developed, the Layer Shift (LS) Detection strategy. As its name implies, this strategy is specifically targeted at determining when a layer shift fault has occurred. For this strategy, sample points just beyond the outer shell of the print and expecting the print to not be detected. Examples of such sample points can be seen in Figure 6(c). When a layer shift fault does occur, the most recently printed layer will be offset from the previous layers, causing the sample point to be shifted over the previous layers resulting in a "print detected" result. Figure 7 illustrates this process.

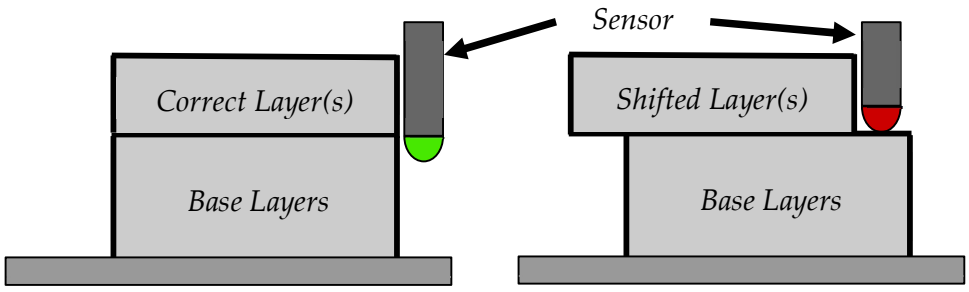


Figure 7. Layer shift detection sampling strategy

A point may be raised that a print which has a layer shift at the same time as a transition from a small cross section in the previous layer to a larger cross section in the current layer may not allow the previous smaller layer to be detected. However, such occurrences can only take place in two situations, which are illustrated in Figure 8. In the first situation, an overhang allows for the sudden change in cross section. In this case, the overhang will require support material to be printed, which will be detected when a

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 December 2021 doi:10.20944/preprints202112.0460.v1

sample is taken after a layer shift. The second case shows a print with a gradual change in the cross section between layers. In this case, there is no support material, but the difference in cross sections is so small, a sample point will still detect the previous layer when a millimeter-level layer shift occurs.

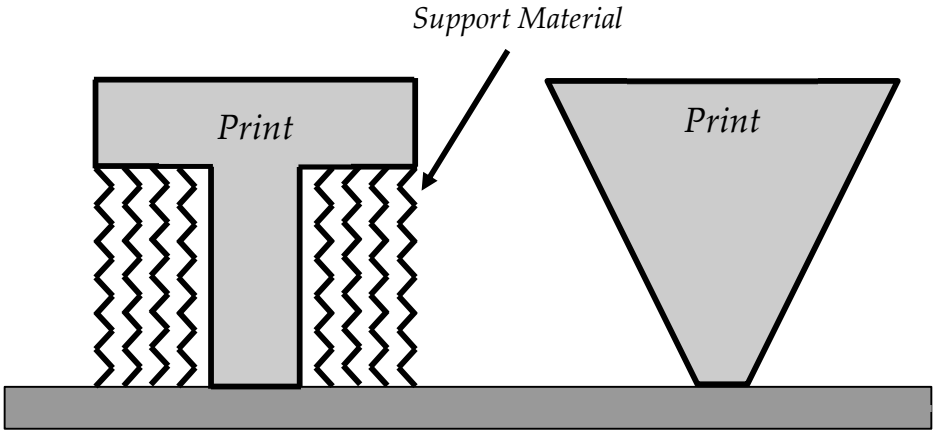


Figure 8. Two edge cases for the layer shift detection.

2.2.4. Fault Detection

Once the sample points have been generated and the user is satisfied, the print can be started, and the fault detection will be handled automatically during the printing process. As the part is being printed, the control software will interrupt the printing process after a layer with sample points has been finished. The program then iterates through the generated sample point coordinates. For each point, a command is generated from the sample points coordinates and the X and Y offset values to account for the difference between the nozzle of the extruder and the MTouch sensor. When the sensor is in position, a measurement is taken, and the results compared to the expected result. If the two values match, the process repeats for the next sample point. If the two values do not match, the print is paused the user is alerted. Once all sample points are checked and no discrepancies have been detected, the print is resumed.

2.3. Actuated Contact Sensor

2.3.1. Sensor Construction

Having developed a methodology to detect faults, a sensor was needed, capable of detecting the presence or absence of the print at specific locations. An actuated contact sensor was chosen for its low-cost and tolerance of external conditions compared to other methods. The following criteria were developed:

1. Total cost is less than $100;
2. Able to be controlled and powered by a Raspberry Pi;
3. Average time for one sample is less than 0.5 seconds;
4. Should not mark the print upon contact;
5. Greater than 95% accuracy.

Criteria 1 and 2 were established to satisfy the original goal of the project to create a low-cost fault detection system. A target price of <$100 was established as this is significantly less than the $1,196 average selling price of desktop 3D printers [2]. A key in making the system low cost is by minimizing the cost of computational hardware by only using a Raspberry Pi to control and power to system. Criteria 3, 4, and 5 were established to ensure the system is minimally intrusive. If the time required to take a sample is too large, using the fault detection system will cause a large increase in the time required to print an object, outweighing any time savings when a fault occurs. Similarly, if

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 December 2021 doi:10.20944/preprints202112.0460.v1

the sensor were to mark the print upon contact, the quality of the print would be diminished making users less likely to use the system. Finally, the sensor must be accurate to minimize false positive and negatives pausing the print when a fault has not occurred.

One commercially available contact sensor (and its derivatives) which meets most of these criteria is the BLTouch. The BLTouch is a bed leveling sensor that uses an actuated probe to contact the print bed and act as an end stop. During the initial development of the MTouch system, the BLTouch was tested to see if it was suitable for use. However, testing revealed that the probe of the BLTouch could become stuck while taking measurements. This was caused by the deployment force of the probe being less than the retraction force. The probe would become wedged between the side of the print and the sensor body during deployment and would not be able to overcome friction when retracting. In addition to becoming stuck alongside the print, the BLTouch was unable to detect when it had become stuck. The printer would then force the probe into the print, bending the probe, as it continued, damaging the BLTouch and the print.

To resolve this issue, a custom sensor was designed with a stronger retraction force and the ability to determine if it became stuck. Figure 9 shows final MTouch sensor design with labels indicating the major components The design starts with a 3D printed bracket that attaches to the extruder of the 3D printer and acts as the attachment point for a solenoid and a hall effect sensor. The solenoid is an Adafruit Mini Push-Pull Solenoid (product ID #2776) and it is responsible for the actuation of the design. The specific model of solenoid was chosen for several reasons. First, its power requirements were within the limits of a Raspberry Pi (rPi). Second, the solenoid is a push-pull configuration with a spring return. A spring return is required as the solenoid needs to actuate down on command but must return to a retracted position when not powered to prevent collisions with the part being printed.

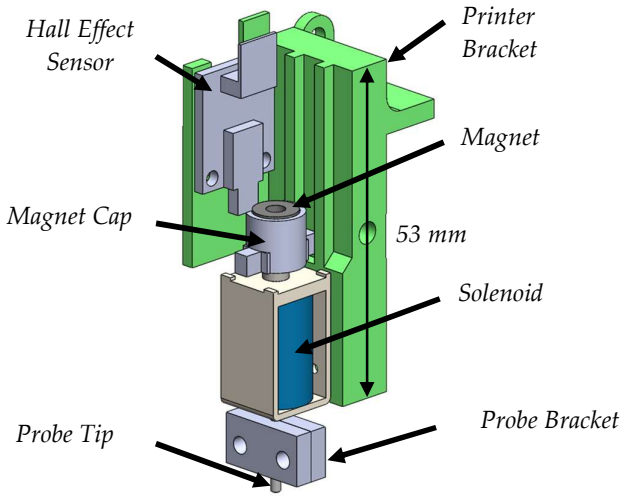


Figure 9. MTouch Sensor

Connected to the solenoid are two additional 3D printed parts, the magnet cap and the probe tip. The magnet cap simply acts as an adaptor between the core of the solenoid and the magnet. The magnet is needed for the hall effect sensor to determine the position of the solenoid during operation. The specific type of magnet used is a Super Magnet Man south pole radial ring magnet (part #RR0060S). This magnet is unique because the entire outer diameter is the south pole of the magnet. A radial magnet is used to eliminate any effect from the solenoid core rotating and changing the alignment between the magnet and hall effect sensor. Additionally, the magnet cap has an extension which sits between two rails on the bracket preventing the magnet from rotating more than a few degrees. The south pole of the magnet was chosen to be on the outer diameter since the type of hall effect sensor used (A3144 hall effect sensor) is more sensitive to the south

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 December 2021 doi:10.20944/preprints202112.0460.v1

pole of a magnet. It is important to note the type of hall effect sensor chosen outputs a binary signal (magnet detected or not detected), as the rPi would be unable to read an analog input without additional hardware. The second 3D printed part, the probe tip, also acts as an adaptor, this time between the solenoid core and a 1/16th inch dowel (McMaster-Carr 98381A415). The dowel is used to reduce the diameter of the probe contacting the printed part, increasing the resolution of detail which can be measured. The solenoid and hall effect sensor are connected to the rPi as shown in the circuit diagram in Figure 10. All these components sum to a cost of $59.93. Since the MTouch sensor includes 3D printed parts and requires some wires, the total unit cost will be slightly higher but well within the $100 target. For comparison, the cost of a BLTouch is $40, but a rPi is also required for operation, brining the cost up to $68.89.

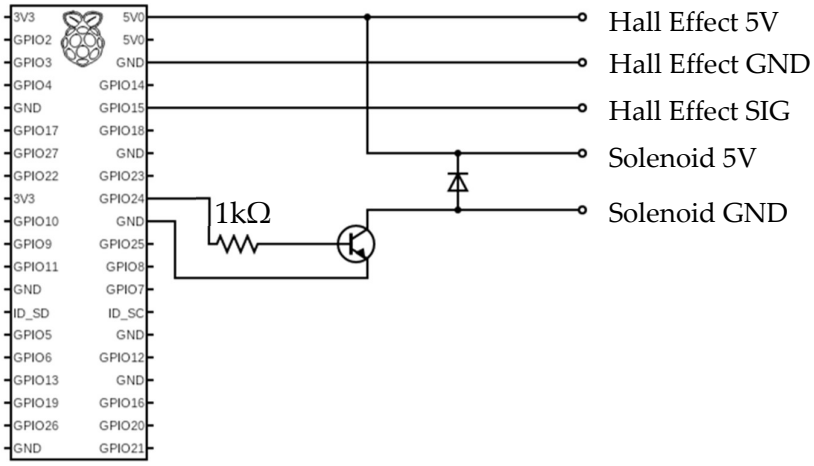
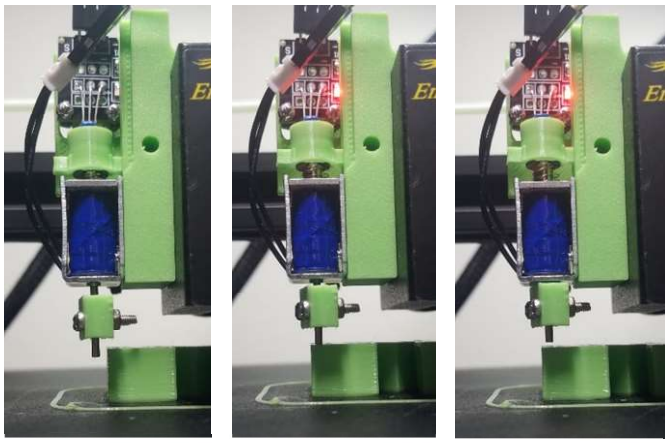


Figure 10. Circuit diagram the connections between the Raspberry Pi, hall effect sensor, and solenoid.

2.3.2. Sensor Operation

The contact sensor determines the presence or absence of the print by actuating the solenoid and using the hall effect sensor to measure the position of the magnet. Images of the various states of the sensor during the measurement process can be seen in Figure 11. The rightmost image in the figure shows the sensor in the stowed position above the intended sample point. In this position the magnet is close enough to the hall sensor for the magnet to be detected (which can be seen by the red light). The middle image shows the sensor when the solenoid is actuated, and the probe contacts the print. The magnet moves away from the hall effect sensor, but the magnet can still be detected. When the rPi reads the signal, from the hall effect sensor, it will know the probe contacted the print. The leftmost image shows the solenoid actuated and the probe not in contact with the print. Since the probe is not in contact with the print, the solenoid is able to fully extend, moving the magnet far enough away from the hall sensor, the magnet cannot be detected. This is read by the rPi which interprets the input as an indication the probe did not contact the print. Note, a hall effect sensor and magnet are not specifically required. A similar effect can be achieved by using other sensors such as optical endstops, contact switches, and capacitance probes. These other sensor may also decrease the total cost of components for a MTouch sensor.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 December 2021 doi:10.20944/preprints202112.0460.v1



(**a**) (**b**) (**c**)

Figure 11. Three images representing the three states of the sensor during use: **(a)** The deployed position not in contact with the print; **(b)** The deployed position in contact with a print; **(c)** The stowed position.

The process of taking a measurement with the sensor is illustrated in a flowchart shown in Figure 12 below. The process starts with the control pushing power to the solenoid. The control program then waits 0.15 seconds for the solenoid to actuate before taking a reading from the hall sensor. The period of 0.15 seconds was found to be the shortest period the position of the magnet could reliably be determined. Next, power cut from the solenoid and another 0.15 seconds is allowed to pass for the solenoid to retract. At this point, a second hall effect sensor reading is taken to confirm the probe was able to retract and is not stuck alongside the print. If the probe is stuck, a subroutine consisting of moving the probe 0.1 mm (a value determined empirically) along the X and Y axes outward from the sample point location to free it. To reduce the rate of false fault detections, a second sample can also be taken whenever there is a difference between the measured result and the expected result. This addition increases the time required for a measurement, but a sensor with high accuracy should not require this very often, increasing the print time minimally while preventing possible false fault detections. As previously mentioned, once a sensor measurement is established, the control software compares the result to the predetermined expected value. If the two match, the printer moves to the next sample point and repeats the measurement process, or the print is resumed. If the results do not match, the print is paused. The user can then check the print and decide between continuing the print and restarting.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 December 2021 doi:10.20944/preprints202112.0460.v1

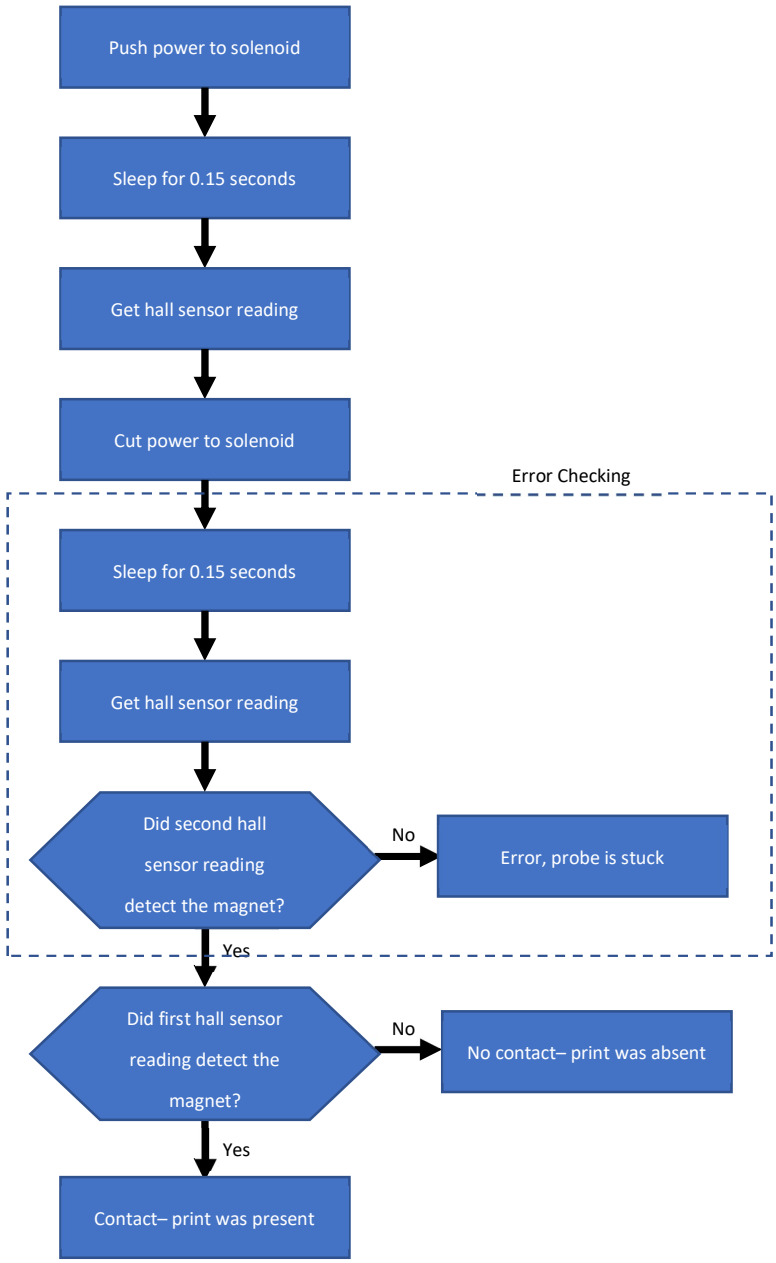


Figure 12. A flowchart illustrating the steps taken to get a measurement with the MTouch sensor.

2.4. Experimental Setup

To test the MTouch sensor and proposed methodology, a fault detection system was implemented on an Ender 3 Pro 3D printer. To control the printer and interface with the MTouch sensor, an Octoprint plugin was created. Octoprint is an open source program that gives users the ability to use a web interface to monitor and control their printers [25]. Octoprint runs on a separate computer from the printer (typically a rPi) and streams commands from the user or from an uploaded Gcode file to a connected device. This method of integration was chosen as Octoprint provides a framework for easy plugin creation and the ability to execute on a rPi.

As the part is manufactured, the plugin will interrupt the print and take control of the printer after a layer with sample points has been completed. This interruption is facilitated by checking the current layer height after a Z-change event, which is broadcast by the main Octoprint software. If the layer height matches that of a layer with sample points,

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 December 2021 doi:10.20944/preprints202112.0460.v1

the main Octoprint thread is paused while another processing thread is created to handle sample point measurements. This thread iterates through the generated sample point coordinates. At each point, a G0 command is generated from the sample points coordinates and the sensor offset values. After the G0 command, a M114 (return current position) command is also sent to the print. This allows the plugin to know when the extruder is in position as the current position message will only be sent after processing the G0 command. Once this message is received, the plugin executes the measurement routine described in Section 2.3.2. The result of the measurement is then compared to the expected result. If the values match, the process repeats with the next sample point, until all sample points are completed at which time the main Octoprint thread will be continued. If the values do not match, a second measurement is taken. If this second measurement confirms the discrepancy, a fault has been detected and a flag is set in the plugin. The main Octoprint thread is resumed which checks the plugin flag and will pause the print.

3. Results

3.1. Sensor Validation Testing

To validate, the MTouch sensor meets the set requirements, several tests were conducted to validate the custom design as well as testing the BLTouch to see how the sensors compare. Tests were conducted to determine the range over which each sensor could become stuck, the rate of false positives and negatives, the accuracy of each sensor, and the measurement period. Each test is described below and is accompanied by its results.

The first test establishes the range of distances from the edge of a print, where the probe of the sensor becomes stuck. The test starts by printing a 15 mm cube with fault detection disabled. Once the cube is complete, the probe is positioned 2 mm outward from an edge of the cube. A measurement is taken, and the probe checked to determine if it became stuck along the side of the print. The state of the probe is recorded (stuck or not stuck), the probe moved 0.1 mm toward the center of the cube, and another measurement taken. This process is repeated until a measurement 2 mm away the edge of the print, inward toward the center of the cube, was taken. Figure 13 shows an illustration of this test.

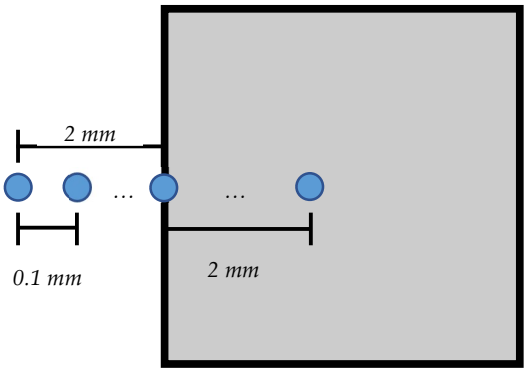


Figure 13. An illustration of the sample points used to test if and where each sensor would become stuck along the side of a print.

The second test measured the rate of false positives and negatives of each sensor. Using the 15 mm cube from the previous test, the rate of false negatives was measured by positioning the sensor at a random point over the print, such that the probe would contact the print, and taking 100 consecutive measurements. The sensor was then moved away from the print, such that the probe would not contact the print, and another 100 consecutive measurements were taken to find the rate of false positives. The results of these two tests are shown in Table 1 below.

Table 1. Results of sensor tests for BLTouch and MTouch

Sensor	Range of Getting Stuck [mm]	Rate of False Positives	Rate of False Negatives
BLTouch	[-0.1, 0.5]	1%	0%
MTouch	(0.0, 0.1]	0%	0%

The results in Table 1 show the MTouch sensor can still become stuck alongside the print but in a significantly narrower range compared to the BLTouch. Table 1 also shows the rates of false positives/negatives are almost identical between the two sensors. The BLTouch did output one false positive reading compared to zero for the MTouch sensor, but this is most likely due to natural variance. Similarly, while the MTouch sensor had a 0% rate of false positives and negatives, this does not mean the sensor to completely immune to false readings. During testing of the entire fault detection system, as described in the next section, there were three instances of false readings being output out of hundreds of measurements. This does indicate that the rate of false readings is very low making the sensor suitable for use.

To further test the accuracy of each sensor in a less artificial case, each sensor was used to take 100 random samples (including points where both the presence and absence of the print is expected) across the cross section of a partially completed print of a 15mm cube with no defects. The accuracy was then calculated as the number of correct measurements divided by the total number of measurements. The test was run a second time with measurements whose results deviated from the expected result, being sampled a second time (or "resampled") to ensure a fault has been accurately been detected. The results of these tests are shown in Table 2. The results show the MTouch has 100% accuracy in determining the presence of absence of the print with and without the second measurement. However, the BLTouch had, at most, 74% accuracy. This is considerable difference compared to the almost 0% rate of false positives and negatives. This difference is due to the probe of the BLTouch occasionally becoming stuck along the side of the cube when sampling near edges, causing an inaccurate measurement.

Table 2. Results of accuracy test for BLTouch and MTouch

Sensor	Accuracy Without Second Measurement	Accuracy With Second Measurement
BLTouch	72%	74%
MTouch	100%	100%

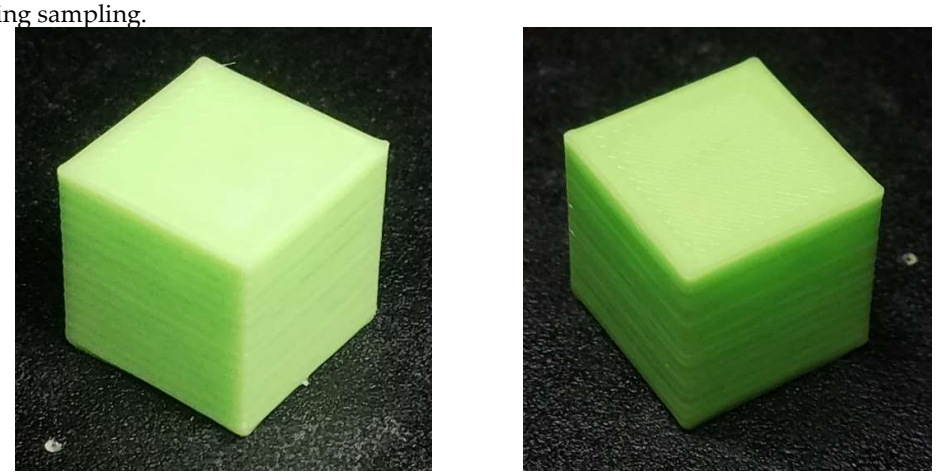
To determine the measurement period, or time needed for a single measurement, for each sensor, 100 consecutive samples were taken, and the total time recorded. The time for a single measurement was then assumed to be the average time per sample. Note, the timing values had to be changed from 0.15 seconds for the MTouch to 0.3 seconds for the BLTouch due to slower probe deployment. The measurement period for the MTouch without the error check (i.e., the second hall sensor reading in Figure 12 to detect a stuck probe) was also determined. The results, in Table 3, show the MTouch sensor has a measurement period 0.378 seconds even with error checking, faster than the BLTouch by 47%. This measurement period is also within the 0.5 seconds established in the sensor requirements.

Table 3. Results of timing tests for BLTouch and MTouch

 doi:10.20944/preprints202112.0460.v1

Sensor	Measurement Period without Error Checking [s]	Measurement Period with Error Checking [s]
BLTouch	0.718	N/A
MTouch	0.212	0.378

Since many other fault detection systems use non-contact sensors, the need to physically interact with a newly printed layer raises concerns about distorting the print at sample points. A final test was conducted to determine if any artifacts would be left on the print by the sampling process. The test consisted of printing a 15mm cube and sampling six random points and each corner of every third layer and the last layer. A print with a relatively small cross-sectional area was chosen to give the layer less time to cool, making the layer the most susceptible to artifacts from the measurement process. Figure 14 shows two cubes, one printed with the sampling process and one printed without. The cube printed with the sampling process shows no signs of being scuffed or marked by the probe during sampling.



(**a**) (**b**)

Figure 14. Two images of the cubes printed with (**a**) and without (**b**) being sampled.

3.2. MTouch System Validation Testing

To validate the MTouch system and methodology of detecting faults, tests were conducted where a fault would be deliberately introduced to a print with the fault detection system in operation. The print layer the fault was introduced and the layer the fault was detected were recorded as well as the time elapsed before detection. The faults tested were layer bed separation, filament runout/jam, and layer shifting. Images of a normal print as well as a print affected by each of these faults can be seen in Figure 15. As a point of comparison, the same tests were conducted with The Spaghetti Detective (TSD) using the default settings and a Logitech C270 Webcam (720p resolution) mounted to the extrusion motor of the Ender 3 Pro using a 3D printed bracket and zip ties. Each test is described below and is accompanied with its results. The sample point generation strategy setting used during all tests are shown in Table 4.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 December 2021 doi:10.20944/preprints202112.0460.v1

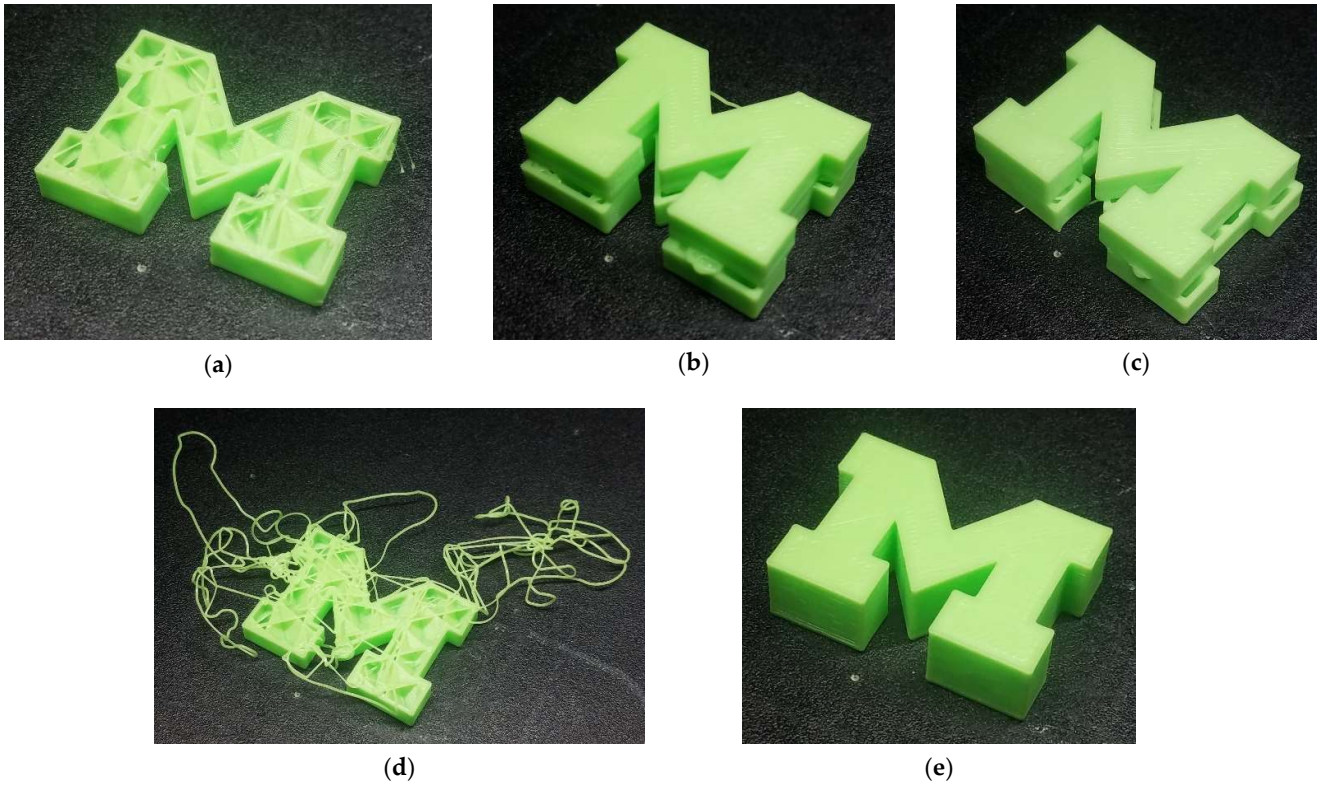


Figure 15. Examples of all the faults tested in the fault detection validation tests: (**a**) Filament Runout/Jam (**b**) Y Axis Layer Shift (**c**) X Axis Layer Shift (**d**) Bed Separation (**e**) No Defects

Table 4. Sample point generation strategy settings used during all tests of the MTouch system.

Layer Spacing	5
Layer Shift Detection	4 points per layer
Random Sampling	1 point per cm^2
Min-Max Sampling	4 points per layer
Inside-Outside Sampling	1 point per cm^2

The first fault detection test consisted of a bed separation fault. To reliably introduce such a fault, the print was paused before layer 27 of 52 began to print and a metal spatula was used to separate the print from the print bed while the part was held in place. The print was then resumed. Layer 27 was chosen as it is toward the middle of the print and is immediately after sample points are measured for layer 26, maximizing the number of layers and time until another layer of samples would be printed (Layer 32). The results of the test, presented in Table 5, shows MTouch detected the fault while sampling the first layer with sample points after the fault was introduced. Additionally, MTouch was able to detect the fault 154 seconds (or 44%) faster than The Spaghetti Detective.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 December 2021 doi:10.20944/preprints202112.0460.v1

Table 5. Results of bed separation fault detection tests for MTouch and TSD when the fault was introduced at layer 27.

Fault Detection System	Layer Fault was Detected	Elapsed Time [s]
TSD	35	349
MTouch	32	195

The second fault detection test consisted of a filament runout/jam fault. This fault was caused by removing filament from the extruder. Again, this fault was introduced at layer 27. The results in Table 6, show the MTouch detecting the fault while sampling the first layer with sample points after the fault was introduced. This is compared to The Spaghetti Detective which was unable to detect the fault. Looking at the fault in Figure 15, this is likely because the fault does not cause the secondary fault of spaghetti failure. The part after the fault is introduced keeps a geometric shape despite producing an unusable part.

Table 6. Results of filament runout/jam fault detection tests for MTouch and TSD when the fault was introduced at layer 27.

Fault Detection System	Layer Fault was Detected	Elapsed Time [s]
TSD	N/A	N/A
MTouch	32	201

The third fault detection test consisted of a layer shift fault. This fault was introduced to the printed part by pausing the print at layer 27, finding the current coordinates of the extruder using an M114 command, and using a G0 command to move the nozzle 1.5 mm either the X or Y direction (both were tested). A G92 command was then used to set the position of the extruder back to the coordinates received from the M114 message response. Table 7 shows the results of this test. Looking at the results, the MTouch was, once again, able to detect the fault while sampling the first layer with sample points after the fault was introduced while The Spaghetti Detective was unable to detect the fault. Similar to the filament runout/jam fault, this is likely due to the part keeping a geometric shape and not forming any spaghetti.

Table 7. Results of layer shift fault detection tests for MTouch and TSD when the fault was introduced at layer 27.

Fault Detection System	Layer X Axis Shift was Detected	Layer Y Axis Shift Was Detected
TSD	N/A	N/A
MTouch	32	32

Since the MTouch system pauses the print at regular intervals to measure the sample points, the time required to complete a print will increase. The last fault detection test consisted of printing a regular print with no faults with and without the MTouch system operating. The time required to complete the print in both scenarios was recorded. As Table 8 shows, the MTouch system increased the print time by 163.2 seconds (8.49%) for a 1922.5 second print. It should be noted this increase in print time can vary greatly with the sample point generation strategy values. For example, if a larger layer spacing or a lower point density is used, the increase in time will be less. However, the system may not detect faults as quickly or accurately, since there are fewer sample points to determine when something has gone wrong.

Table 8. Results of print time test while the MTouch system is enabled and disabled.

MTouch System Enabled/Disabled	Time Required for Print [s]
Enabled	2085.7
Disabled	1922.5

4. Discussion, Conclusions and Future Work

This paper has presented a low-cost, robust and versatile system for detecting millimeter scale part defects in desktop FFF 3D printers. The system works by using a custom developed actuated contact sensor to determine the presence or absence of the print at specific locations and comparing the results to pre-calculated expected values. Validation tests were conducted on the developed sensor and fault detection system to show there were operating within the requirements set. From the results of the sensor validation tests, it was shown the actuated contact sensor met all the criteria. During the design of the sensor, the total cost was kept under the target price of $100 and the sensor is able to be controlled and powered by a Raspberry Pi. The validation tests show the average sample period of the sensor is 0.378 seconds with an accuracy of 100% (during accuracy testing), meeting both requirements for measurement period and accuracy. The validation tests also showed the sensor does not mark or mar the surface of the print while taking a sample. The results of the system validation tests also show that the MTouch system can detect millimeter-scale major faults that are undetectable using The Spaghetti Detective while minimally increasing the time required to complete a print.

Finally, there are a few areas of potential future work with the MTouch fault detection system. The first area is shifting the sample point generation from Octoprint to a slicer like Cura [26]. This would save a significant amount of time as the model recreation process can be skipped. The second area of future work is using the fault detection system to autonomously collect data to train a neural network to predict faults from the Gcode. By predicting a fault ahead of time this would allow proactive steps to be taken to prevent printing part from becoming unsuitable for use in the first place. This is a topic of ongoing work at the Smart and Sustainable Automation (S2A) Lab at the University of Michigan.

6. Patents

The Regents of the University of Michigan have filed a provisional patent for the MTouch fault detection system.

Author Contributions: Conceptualization, S.A., C.O.; methodology, S.A., K.W., Z.E., N.C.; software, S.A., K.W., Z.E., N.C.; validation, S.A., K.W., Z.E., N.C.; formal analysis, S.A., K.W.; resources, C.O.; data curation, S.A.; writing—original draft preparation, S.A.; writing—review and editing, C.O.; visualization, S.A.; supervision, C.O.; project administration, C.O..; funding acquisition, C.O. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The source code for the MTouch Octoprint plugin can be found at: https://doi.org/10.5281/zenodo.5796404. The CAD files for the MTouch system can be found at: https://doi.org/10.5281/zenodo.5796477

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

References

1. Sculpteo, Inc., The state of 3D printing: the data you need to understand the 3D printing world and build your 3D printing strategy, San Francisco, CA, 2021.
2. Wohlers, T., Wohlers Report 2020: 3D Printing and Additive Manufacturing: Global State of the Industry. In Technical report. Wohlers Associates, 2020.
3. Song, R.; Telenko, C, Causes of Desktop FDM Fabrication Failures in an Open Studio Environment, Procedia CIRP 2019, 80, 494-499.
4. Print quality troubleshooting, Available online: www.simplify3d.com/support/print-quality-troubleshooting/ (accessed November 15, 2021).
5. Spaghetti monster, Available online: https://help.prusa3d.com/en/article/spaghetti-monster_1999/ (accessed November 15, 2021).
6. Spaghetti detective octoprint guide, Available online: https://all3dp.com/2/spaghetti-detective-octoprint-guide/ (accessed November 15, 2021).
7. Duan M.; Yoon D.; Okwudire C.E., A limited-preview filtered B-spline approach to tracking control–With application to vibration-induced error compensation of a 3D printer, Mechatronics, 2018, 56, 287-296.
8. Fu Y.; Downey A.; Yuan L.; Pratt, A.; Balogun Y., In situ monitoring for fused filament fabrication process: A review, Addit. Manuf. 2021, 38.
9. Delli, U.; Chang, S., Automated Process Monitoring in 3D Printing Using Supervised Machine Learning, Procedia Manuf. 2018, 26, 865-870.
10. Khan, M.F.; Alam, A.; Siddiqui, M.A.; Alam, M.S.; Rafat, Y.; Salik, N.; Al-Saidan, I., Real-time defect detection in 3D printing using machine learning, Mater. Today: Proc. 2021, 42, 521-528.
11. Holzmond, O.; Li, X., In situ real time defect detection of 3D printed parts, Addit. Manuf. 2017, 17, 135-142.
12. Cheng, Y.; Jafari, M. A., Vision-Based Online Process Control in Manufacturing Applications, IEEE Trans. Autom. Sci. Eng. 2008, 5, 140-153.
13. Nuchitprasitchai, S., An algorithm for reconstructing three-dimensional images from overlapping two-dimensional intensity measurements with relaxed camera positioning requirements, with application to additive manufacturing, PhD thesis, Michigan Technological University, Houghton, MI, 2017.
14. Nuchitprasitchai, S.; Roggemann, M.C.; Pearce, J.M., Three hundred and sixty degree real-time monitoring of 3-D printing using computer analysis of two camera views, J. Manuf. Mater. Process. 2017.
15. The Spaghetti Detective, Available online: https://www.thespaghettidetective.com/docs/optimal-camera-setup/ (accessed November 15th, 2021).
16. Youtube, Available online: https://youtu.be/VvsM8Np4Qkg (accessed November 15th, 2021).
17. Lin, W.; Shen, H.; Fu, J.; Wu, S., Online quality monitoring in material extrusion additive manufacturing processes based on laser scanning technology, Precis. Eng. 2019, 60, 76-84.
18. Khanzadeh, M.; Rao, P.; Jafari-Marandi, R.; Smith, B.K.; Tschopp, M. A.; Bian, L., Quantifying Geometric Accuracy With Unsupervised Machine Learning: Using Self-Organizing Map on Fused Filament Fabrication Additive Manufacturing Parts, J. Manuf. Sci. Eng. 2018; 140.
19. Tootooni, M.S.; Dsouza, A.; Donovan, R.; Rao, P. K.; Kong, Z. J.; Borgesen, P., Classifying the dimensional variation in additive manufactured parts from laser-scanned three-dimensional point cloud data using machine learning approaches, J Manuf Sci Eng. 2017, 139.
20. Li Z.; Zhang, Z.; Shi, J.; Wu, D., Prediction of surface roughness in extrusion-based additive manufacturing with machine learning, Robot Comput Integr Manuf. 2019, 57, 488-495.

21. Li Y.; Zhao W.; Li Q.; Wang T.; Want G.; In-situ monitoring and diagnosing for fused filament fabrication process based on vibration sensors, J. Sens. 2019.
22. Liu, C.; Law, A.C.C.; Roberson, D.; Kong, Z.J., Image analysis-based closed loop quality control for additive manufacturing with fused filament fabrication, J. Manuf. Syst. 2019, 51, 75-86.
23. Rao, P.K.; Liu, J.; Roberson, D.; Kong, Z.; Williams, C.. Online Real-Time Quality Monitoring in Additive Manufacturing Processes Using Heterogeneous Sensors, J. Manuf. Sci. Eng. 2015, 137.
24. Li, Z., Zhang, Z., Shi, J., Wu, D., Prediction of surface roughness in extrusion-based additive manufacturing with machine learning, Robot Comput Integr Manuf 2019, 57, 488-495.
25. Octoprint, Available online: https://octoprint.org/ (accessed December 7th, 2021).
26. Ultimaker, Available online: https://ultimaker.com/software/ultimaker-cura (accessed November 15th, 2021).